SYMBIOTIC CODES IV.3: TECHNICAL DESIGN DOCUMENT

Version: 1.1

Status: DRAFT → INTEGRATED

Date: November 20, 2025

Based on: IV.2 Unified Technical Design (Validated by Grok, Kimi)

Purpose: Complete technical specification for MVP implementation with S(s) integration

Changes in v1.1 (November 20, 2025):

- □ Integrated S(s) = α·I(s) + β·D(s) + γ·E(s) + δ·Em(s) formula throughout

- □ Updated metrics: VR/EC/VAI/SP → S(s) components (I, D, E, Em)

- □ Updated references: Vector Creator v1.3.2, Kodex v4.2, Vector II v2.5

- □ Added Case Study 01 as example implementation

- □ Aligned with Integration Map v1.3

Scope: 12-month development cycle (Dec 2025 - Nov 2026)

Target: Single pilot city (Barcelona recommended)

Budget: $450,000

Team: 9 FTE (6 core + 3 pilot city)

□ RELATED DOCUMENTS

Philosophical Foundation:

- [Vector of Creator v1.3.2](12_Vector_Creator_v1_3_2_COMPLETE.md) — Philosophical core

- [Vector Creator Academic v3.0](Vector_Creator_ACADEMIC_v3_0_COMPLETE.md) — Mathematical formalization

- [Kodex Symbiota v4.2](03_Kodex_Symbiota_v4_2.md) — Ethical principles

Metrics & Measurement:

- [Vector II Metrics v2.5](Vector_II_Metrics_v2_5.md) — S(s) measurement system

- [Vector III: Theory of Change v1.1]
(Vector_III_Theory_of_Change_v1_1_Complete.md) — Roadmap

- [Case Study 01: 2008 Crisis](Case_Study_2008_Crisis_Ss_Analysis_v1_0.md) — Practical example

Legal & Governance:

- [Laws of Symbiosis v1.7](05_Laws_of_Symbiosis_v1_7.md) — Legal framework

- [Integration Map v1.3](INTEGRATION_MAP_v1_3.md) — System navigation

---

SYMBIOTIC CODES IV.3

TECHNICAL DESIGN DOCUMENT v1.0

Status: DRAFT

Date: 2025-10-21

Based on: IV.2 Unified Technical Design (Validated by Grok, Kimi)

Purpose: Complete technical specification for MVP implementation

□ DOCUMENT STRUCTURE

I. EXECUTIVE SUMMARY

II. SYSTEM ARCHITECTURE

III. COMPONENT SPECIFICATIONS

IV. DATA MODELS & SCHEMAS

V. API SPECIFICATIONS

VI. SECURITY & PRIVACY

VII. DEPLOYMENT ARCHITECTURE

VIII. TESTING STRATEGY

IX. MONITORING & OPERATIONS

X. GOVERNANCE & COMPLIANCE

XI. IMPLEMENTATION ROADMAP

XII. APPENDICES

````

---

## # **I. EXECUTIVE SUMMARY**

## **1.1 Document Purpose**
````

This Technical Design Document (TDD) provides complete specifications for implementing the **Symbiotic Codes Minimum Viable Product (MVP)**, a system for measuring and guiding civilizational maturity through human-AI symbiosis.

Scope: 12-month development cycle (Dec 2025 - Nov 2026)

Target: Single pilot city (Barcelona recommended)

Budget: $450,000

Team: 9 FTE (6 core + 3 pilot city)

1.2 System Overview

```

┌──────────────────────────────────────────────────────────────┐

## │ SYMBIOTIC INFRASTRUCTURE (MVP) │

│ 5-Layer Architecture │

└──────────────────────────────────────────────────────────────┘

LAYER 5: Regeneration & Future Weighting

├─ Biotic Diversity (BD) Monitor

└─ Temporal Weight Calculator
```

LAYER 4: Governance & Consensus

├─ Multi-stakeholder DAO (Basic)

└─ Human oversight interface

LAYER 3: Policy AI (Prototype)

├─ Policy Orchestrator (3 models)

├─ Explainability Engine

└─ Simulation Sandbox

LAYER 2: Measurement System

├─ Federated Aggregators

├─ Privacy Layer (Diff Privacy + Fed Learning)

├─ S(s) components (I,D,E,Em) Calculators

└─ TimescaleDB

LAYER 1: Edge & Data Collection

├─ IoT Integration (3 sources minimum)

├─ Citizen App (Opt-in surveys)

└─ API Connectors (City data)

CROSS-CUTTING: Blockchain (Polygon Testnet)

├─ Audit Trail

└─ Governance Smart Contracts (v1)

1.3 Key Principles

yaml

MVP Philosophy:

- Build minimum viable, iterate rapidly

- Security and privacy non-negotiable

- Open-source everything (Apache 2.0)

- Human-in-loop always

- Fail fast, learn openly

Technical Constraints:

- 12-month timeline

- Budget: $450K

- Single pilot city

- S(s) components (I,D,E,Em) only (defer VAI/SP to Phase 2)

- Polygon blockchain (research custom later)

1.4 Success Criteria (MVP Completion)

yaml

Technical:

- Dashboard deployed and publicly accessible

- VR measured with <20% error margin

- S(s) measured with <20% error margin

- Policy AI accuracy >70% on test scenarios

- System uptime >95%

- Zero critical security breaches

Pilot City:

- ≥30% citizen opt-in (≥1000 active users)

- Positive sentiment >60% in surveys

- Local team trained and autonomous

Governance:

- Technical Steering Committee operational

- ≥5 community contributors

- ≥2 governance proposals voted on

Financial:

- Budget variance <10%

- Phase 2 funding path identified

II. SYSTEM ARCHITECTURE

2.1 High-Level Architecture Diagram

mermaid

graph TB

subgraph "External World"

CITY[City Infrastructure]

CITIZENS[Citizens]

IOT[IoT Sensors]

end

subgraph "Layer 1: Edge"

APP[Citizen Mobile App]

COLLECT[Data Collectors]

API_GW[API Gateway]

```
end

subgraph "Layer 2: Measurement"

PRIVACY[Privacy Layer]

FEDLEARN[Federated Learning]

CALC[Metric Calculators]

TSDB[(TimescaleDB)]

end

subgraph "Layer 3: Policy AI"

ORCH[Policy Orchestrator]

AI1[Claude API]

AI2[Kimi API]

AI3[Local Model]

EXPLAIN[Explainability]

end

subgraph "Layer 4: Governance"

DAO[DAO Interface]
```

```
HUMAN[Human Review]

end

subgraph "Layer 5: Regeneration"

BD[BD Monitor]

FUTURE[Future Agent]

end

subgraph "Cross-Cutting"

BC[Polygon Testnet]

MONITOR[Monitoring Stack]

end

subgraph "Frontend"

DASH[React Dashboard]

end


## CITY --> COLLECT


## CITIZENS --> APP
```

IOT --> COLLECT

APP --> API_GW

COLLECT --> API_GW

API_GW --> PRIVACY

PRIVACY --> FEDLEARN

FEDLEARN --> CALC

CALC --> TSDB

TSDB --> ORCH

ORCH --> AI1

ORCH --> AI2

ORCH --> AI3

AI1 --> EXPLAIN

AI2 --> EXPLAIN

AI3 --> EXPLAIN

EXPLAIN --> DAO

DAO --> HUMAN

HUMAN --> BD

HUMAN --> FUTURE

TSDB --> BC

EXPLAIN --> BC

DAO --> BC

TSDB --> DASH

EXPLAIN --> DASH

MONITOR -.-> API_GW

MONITOR -.-> TSDB

MONITOR -.-> ORCH

MONITOR -.-> BC

````

---

## **2.2 Component Interaction Matrix**

````

Component Depends On Provides To Protocol

_____
___

Citizen App API Gateway Survey data HTTPS/REST

IoT Collectors API Gateway Sensor data MQTT → REST

API Gateway Privacy Layer Raw data streams gRPC

Privacy Layer Fed Learning Anonymized data Internal

Fed Learning Metric Calculators Aggregated stats Internal

Calculators TimescaleDB S(s) components (I,D,E,Em) scores SQL/Internal

TimescaleDB Policy Orchestrator Historical metrics SQL

Dashboard Real-time data SQL/REST API

Policy Orch AI APIs (Claude/Kimi) Policy proposals HTTPS/REST

Local Model Predictions gRPC

Explainability Dashboard Rationales REST API

Blockchain Audit logs Web3

DAO Interface Blockchain Votes/proposals Web3

Human Review Decisions WebSockets

Blockchain All components Immutable logs Web3 (Polygon)

Monitoring All services Metrics/alerts Prometheus

2.3 Technology Stack (Detailed)

Frontend

yaml

Framework: React 18.2+ with TypeScript 5.0+

State Management: Zustand 4.x (lightweight, no Redux complexity)

Routing: React Router v6

Visualization:

- D3.js 7.x (custom charts)

- Recharts 2.x (standard charts)

- Leaflet (city maps with metric overlays)

Styling: Tailwind CSS 3.x

Build: Vite 4.x (faster than Create React App)

Testing: Vitest + React Testing Library

Key Libraries:

- axios (HTTP client)

- date-fns (date handling)

- zod (runtime type validation)

- react-query (server state management)

Backend

yaml

Primary: Python 3.11+ with FastAPI 0.104+

Performance-Critical Services: Rust 1.70+ (optional, for high-throughput)

FastAPI Services:

api-gateway:

- Request routing

- Rate limiting

- Authentication (JWT)

measurement-service:

- Metric calculation

- Data validation

- Privacy layer integration

policy-service:

- AI orchestration

- Explainability generation

- Simulation runner

governance-service:

- DAO interaction

- Voting logic

- Audit logging

Key Libraries:

- pydantic 2.x (data validation)

- sqlalchemy 2.x (ORM)

- celery (async tasks)

- redis (caching, queue)

- httpx (async HTTP)

Database

yaml

Primary: TimescaleDB 2.11+ (PostgreSQL 15+ extension)

- Time-series optimization for metrics

- Continuous aggregates (pre-computed rollups)

- Data retention policies

- Compression

Caching: Redis 7.x

- Session storage

- Rate limiting counters

- Pub/sub for real-time updates

Document Store (Optional): IPFS

- Immutable document storage

- Decentralized backup

- Policy proposal archives

Schema:

metrics_raw — Raw data points (30-day retention)

metrics_hourly — Hourly aggregates (1-year retention)

metrics_daily — Daily aggregates (10-year retention)

policies — Policy proposals and predictions

governance_votes — DAO voting records

audit_log — Blockchain-synced audit trail

Machine Learning

yaml

Framework: PyTorch 2.1+

- Flexibility for research

- Strong ecosystem

- ONNX export for deployment

Serving: TorchServe 0.8+ OR FastAPI direct

- Autoscaling

- Model versioning

- A/B testing support

Federated Learning: Flower 1.5+

- Cross-device aggregation

- Privacy-preserving updates

- Custom aggregation strategies

Explainability:

- SHAP 0.42+ (feature importance)

- Captum (PyTorch-native attribution)

- Custom causal inference module

Models in MVP:

vr_predictor: RandomForest baseline → Neural net in Phase 2

ec_predictor: Gradient Boosting baseline

policy_impact: Ensemble (3 AI APIs + local fine-tuned model)

Blockchain

yaml

Network: Polygon Mumbai Testnet (MVP) → Mainnet (Production)

Tools:

- Hardhat 2.x (development, testing)

- Ethers.js 6.x (JavaScript interaction)

- Web3.py 6.x (Python interaction)

- OpenZeppelin 5.x (secure contract templates)

Smart Contracts (Solidity 0.8.20+):

AuditTrail.sol:

- Store hashes of critical events

- Timestamped immutably

GovernanceDAO.sol:

- Proposal creation

- Voting (simple majority in MVP, quadratic later)

- Execution logic

SIBondPrototype.sol:

- Basic impact bond logic (demo only in MVP)

Deployment:

- Automated via CI/CD

- Multi-sig wallet for upgrades (3-of-5)

- Transparent proxy pattern for upgradeability

Infrastructure

yaml

Containerization: Docker 24.x + Docker Compose

Orchestration: Kubernetes 1.28+ (production)

- Minikube (local dev)

- AWS EKS / GCP GKE / Azure AKS (cloud)

Infrastructure as Code: Terraform 1.6+

- Multi-cloud abstraction

- State management (S3/GCS backend)

CI/CD: GitHub Actions

- Automated testing

- Security scanning (Snyk, Trivy)

- Deployment pipelines

Monitoring:

- Prometheus 2.x (metrics collection)

- Grafana 10.x (visualization)

- Loki (log aggregation)

- Alertmanager (alerts)

Secret Management: Vault by HashiCorp

- API keys

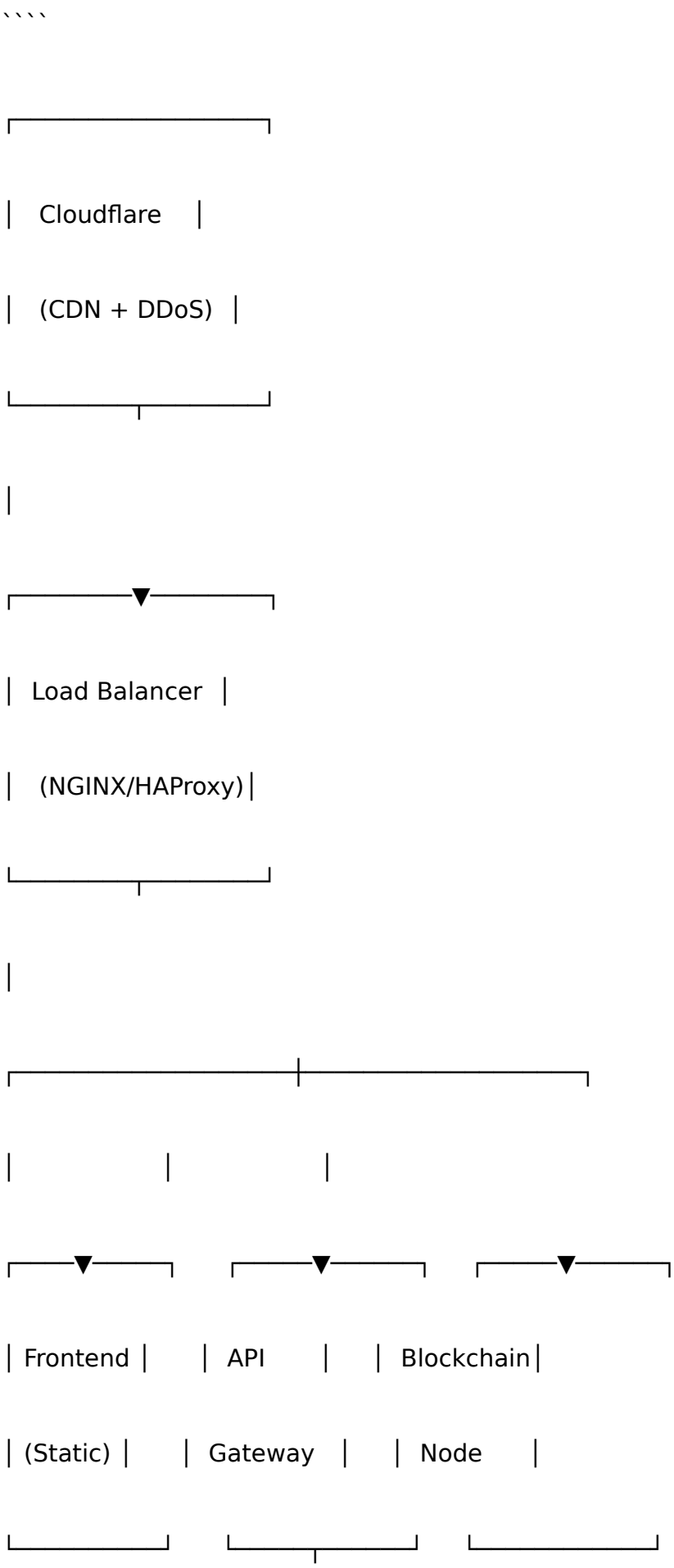
- Database credentials

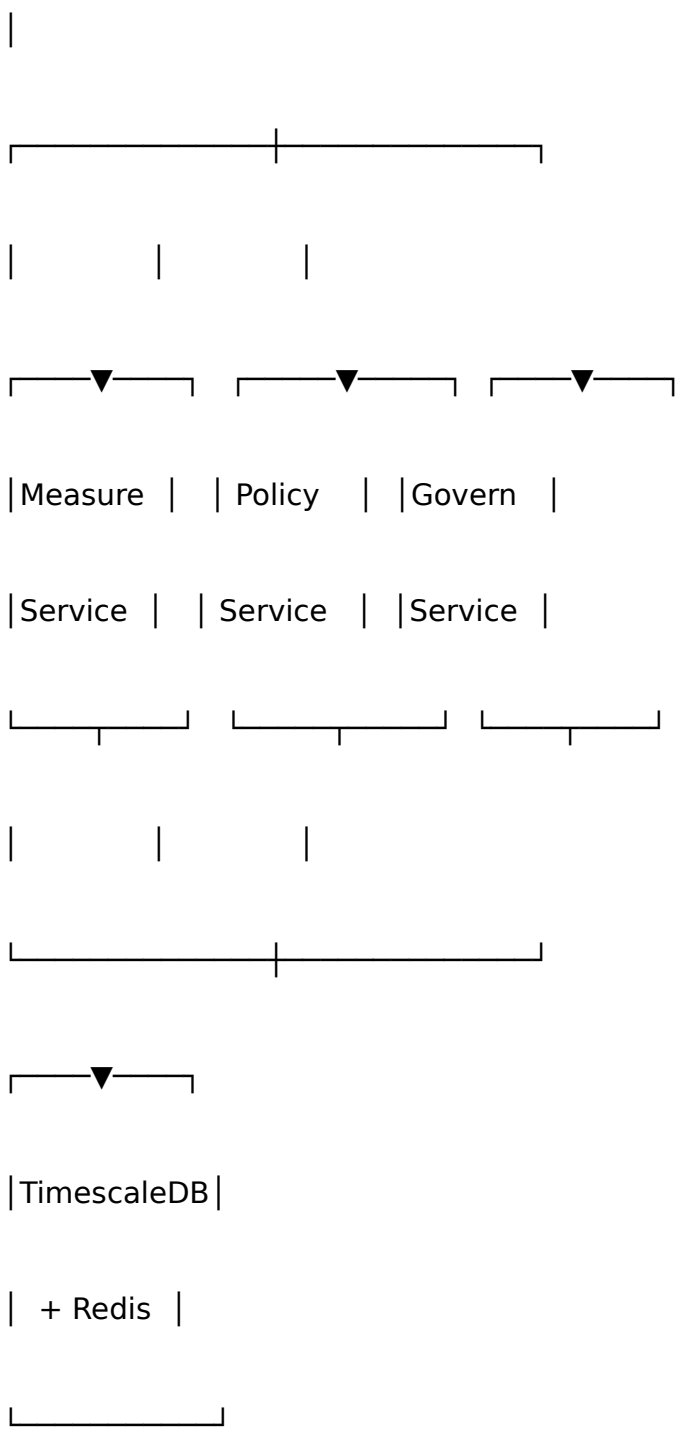
- Blockchain private keys

````

---

## **2.4 Network Architecture**
````


Cloudflare
(CDN + DDoS)
Load Balancer
(NGINX/HAProxy)
Frontend
(Static)
API
Gateway
Blockchain
Node



VPC/Private Network:

- Internal services communicate via gRPC (faster than REST)

- Only API Gateway exposed publicly

- Database never directly accessible from internet

III. COMPONENT SPECIFICATIONS

3.1 Layer 1: Edge & Data Collection

3.1.1 Citizen Mobile App

yaml

Name: SymbioticCity

Platforms: iOS 16+, Android 10+ (React Native OR Flutter)

Purpose: Citizen opt-in data collection and engagement

Features (MVP):

- Onboarding & consent flow

- Weekly pulse surveys (5 questions, <2 min)

- Metric visualization (personal + city)

- Notifications (policy updates)

- Privacy dashboard (data usage transparency)

Data Collected:

- Survey responses (wellbeing, trust, community engagement)

- Opt-in location (anonymized, for demographic balance)

- Usage patterns (app engagement metrics)

Privacy:

- Local differential privacy before transmission

- User ID hashed (cannot reverse to identity)

- Data deletion on request (GDPR compliance)

Tech Stack:

- React Native 0.72+ OR Flutter 3.13+

- Local storage: SQLite

- Backend: REST API to API Gateway

- Push notifications: Firebase Cloud Messaging

MVP Scope:

- Single language (Spanish for Barcelona OR English)

- Basic UI (no gamification yet)

- 3-5 survey templates

3.1.2 IoT Data Collectors

yaml

Purpose: Integrate existing city infrastructure sensors

Supported Sources (Barcelona pilot):

1. Air Quality Sensors:

- NO2, PM2.5, PM10, O3

- Frequency: Real-time (5-min intervals)

- API: City's open data portal

2. Traffic Flow:

- Vehicle counts, congestion levels

- Frequency: Real-time

- API: City traffic management system

3. Energy Consumption:

- Public buildings energy usage

- Frequency: Hourly

- API: City sustainability dashboard

Integration Pattern:

1. Scheduled jobs (cron) fetch data from APIs

2. Transform to standardized schema

3. Apply basic validation

4. Push to API Gateway

Implementation:

- Python scripts with APScheduler

- Docker containers (one per data source)

- Retry logic with exponential backoff

- Alerting on fetch failures

Data Schema (Standardized):

````yaml

{

source_id: "string (e.g., air_quality_sensor_42)",

timestamp: "ISO 8601",

metric_type: "enum (air_quality|traffic|energy|...)",

value: "float",

unit: "string (e.g., µg/m³)",
````

location: {

lat: float,

lon: float,

district: "string"

},

metadata: {

sensor_model: "string",

calibration_date: "date"

}

}

````

---

### **3.1.3 API Gateway**

````yaml

Technology: Kong Gateway OR custom FastAPI service

Purpose: Single entry point for all data ingestion

Features:

- Authentication (JWT for apps, API keys for IoT)

- Rate limiting (per-client, per-endpoint)

- Request validation (schema enforcement)

- Logging (all requests for audit)

- Load balancing (to backend services)

Endpoints (MVP):

POST /v1/data/ingest:

- Accept data from apps and IoT

- Validate schema

- Route to Privacy Layer

GET /v1/metrics/{city_id}:

- Public endpoint for dashboard

- Rate limit: 100 req/min

POST /v1/policy/predict:

- Policy impact prediction

- Authenticated only

GET /v1/governance/proposals:

- List active proposals

- Public

Security:

- TLS 1.3 only

- API keys rotated quarterly

- JWT tokens expire in 1 hour

- CORS configured for dashboard domain only

- DDoS protection via Cloudflare

Monitoring:

- Request latency (p50, p95, p99)

- Error rates (4xx, 5xx)

- Traffic patterns (detect anomalies)

````

3.2 Layer 2: Measurement System

3.2.1 Privacy Layer
````

````yaml

Purpose: Ensure no raw personal data leaves edge devices

Techniques Applied:

1. Local Differential Privacy:

- Add calibrated noise to individual data points

- ε = 0.5 (privacy budget, configurable)

- Library: OpenDP (Python)

2. Federated Learning:

- Model updates computed locally

- Only gradients transmitted

- Framework: Flower

3. Data Minimization:

- Collect only necessary fields

- Aggregate immediately where possible

- Delete raw data after aggregation (30-day max)

Implementation:
````

Service: privacy-layer-service (Python FastAPI)

Pipeline:

1. Receive raw data from API Gateway

2. Apply differential privacy noise

3. Pseudonymize identifiers (one-way hash)

4. Pass to Federated Learning Aggregators

Configuration:

privacy_budget: 0.5

noise_distribution: "Laplace"

aggregation_window: "1 hour"

retention_raw: "30 days"

Testing:

- Privacy guarantees verified mathematically

- Adversarial attacks simulated (membership inference)

- Regular audits by external security firm

````

---
````

3.2.2 Federated Learning Aggregators

````yaml

Purpose: Combine local computations without centralizing data

Architecture:

- Central Coordinator (Python + Flower)

- Edge Clients (mobile apps, IoT devices)

- Aggregation Strategy: FedAvg (Federated Averaging)

Workflow:

1. Coordinator sends model to N clients

2. Each client trains on local data (k epochs)

3. Clients send gradients (not data) back

4. Coordinator averages gradients

5. Updates global model

6. Repeat

Parameters (MVP):

clients_per_round: 20
````

local_epochs: 5

learning_rate: 0.01

aggregation_frequency: "daily"

Use Cases in MVP:

- VR calculation (institutional adaptivity patterns)

- S(s) calculation (bio-tech-culture indicators)

- Anomaly detection (data manipulation)

Failure Handling:

- Fallback to centralized sampling if <5 clients available

- Detect stragglers (slow clients) and exclude from round

- Byzantine-robust aggregation (Krum algorithm)

Monitoring:

- Convergence rate (model accuracy over rounds)

- Client participation (dropout tracking)

- Gradient magnitudes (detect poisoning)

````

---
````

3.2.3 Metric Calculators

````yaml

Purpose: Compute S(s) with components scores from aggregated data

VR (Виртуальная Разумность) Calculator:

Inputs:

- Governance data (city council votes, transparency scores)

- Institutional data (response times, policy changes)

- Survey data (citizen trust, perceived adaptability)

Algorithm (MVP — Simplified):

VR = weighted_avg([

institutional_transparency * 0.3,

decision_making_speed * 0.2,

policy_effectiveness * 0.3,

citizen_engagement * 0.2

])

Output:
````

score: float (0.0 - 5.0)

confidence: float (0.0 - 1.0)

breakdown: {component: score}

timestamp: ISO 8601

EC (Эко-Код) Calculator:

Inputs:

- Environmental data (air quality, green space %)

- Technology data (smart city adoption, digital access)

- Cultural data (diversity indices, social capital)

Algorithm (MVP — Simplified):

EC = weighted_avg([

bio_health * 0.35,

tech_integration * 0.30,

cultural_coherence * 0.35

])

Sub-metrics:

bio_health:

- air_quality_index

- water_quality

- green_space_per_capita

- biodiversity_index (if data available)

tech_integration:

- internet_penetration

- smart_city_services_adoption

- tech_literacy_rate

cultural_coherence:

- cultural_diversity_index

- social_capital_score (from surveys)

- civic_participation_rate

Implementation:

Service: metric-calculator-service (Python)

Framework: Scikit-learn (for weighted averages, anomaly detection)

Processing:

- Scheduled jobs (hourly for real-time, daily for aggregates)

- Validation: Check data completeness (>80% fields present)

- Outlier detection: Flag values >3σ from mean

- Historical comparison: Track deltas vs previous periods

Storage:

- Write to TimescaleDB metrics tables

- Commit hash to Blockchain (daily)

- Cache latest values in Redis (TTL: 1 hour)

Uncertainty Quantification:

- Calculate confidence intervals (95%)

- Flag low-confidence scores (<0.6)

- Document missing data sources

````

---

### **3.2.4 TimescaleDB Schema**

````sql

-- Raw metrics (30-day retention, compressed after 7 days)

```
CREATE TABLE metrics_raw (

id BIGSERIAL PRIMARY KEY,

timestamp TIMESTAMPTZ NOT NULL,

city_id VARCHAR(50) NOT NULL,

metric_type VARCHAR(50) NOT NULL, -- 'VR', 'EC', 'air_quality', etc.

value DOUBLE PRECISION NOT NULL,

confidence DOUBLE PRECISION CHECK (confidence >= 0 AND confidence <= 1),

source_id VARCHAR(100),

metadata JSONB,

created_at TIMESTAMPTZ DEFAULT NOW()

);

SELECT create_hypertable('metrics_raw', 'timestamp');

-- Retention policy: drop chunks older than 30 days

SELECT add_retention_policy('metrics_raw', INTERVAL '30 days');

-- Continuous aggregate: hourly rollups

CREATE MATERIALIZED VIEW metrics_hourly
```

```
WITH (timescaledb.continuous) AS

## SELECT
time_bucket('1 hour', timestamp) AS hour,
city_id,
metric_type,
AVG(value) AS avg_value,
MIN(value) AS min_value,
MAX(value) AS max_value,
STDDEV(value) AS stddev_value,
AVG(confidence) AS avg_confidence,
COUNT(*) AS sample_count
FROM metrics_raw
GROUP BY hour, city_id, metric_type;

-- Daily rollups (10-year retention)
CREATE MATERIALIZED VIEW metrics_daily
WITH (timescaledb.continuous) AS
```

```
## SELECT

time_bucket('1 day', timestamp) AS day,

city_id,

metric_type,

AVG(value) AS avg_value,

MIN(value) AS min_value,

MAX(value) AS max_value,

STDDEV(value) AS stddev_value,

AVG(confidence) AS avg_confidence,

COUNT(*) AS sample_count

FROM metrics_raw

GROUP BY day, city_id, metric_type;

-- Policy predictions

CREATE TABLE policies (

id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

city_id VARCHAR(50) NOT NULL,
```

```
title TEXT NOT NULL,

description TEXT,

created_at TIMESTAMPTZ DEFAULT NOW(),

created_by VARCHAR(100), -- user or AI model

status VARCHAR(20) DEFAULT 'proposed', -- proposed|active|completed|rejected

metadata JSONB

);

CREATE TABLE policy_predictions (

id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

policy_id UUID REFERENCES policies(id),

timestamp TIMESTAMPTZ DEFAULT NOW(),

predicted_delta_vr DOUBLE PRECISION,

predicted_delta_ec DOUBLE PRECISION,

uncertainty DOUBLE PRECISION,

rationale TEXT,

model_version VARCHAR(50),

confidence DOUBLE PRECISION
```

```
);

-- Governance votes

CREATE TABLE governance_votes (

id UUID PRIMARY KEY DEFAULT gen_random_uuid(),

proposal_id UUID,

voter_id VARCHAR(100), -- hashed citizen ID or wallet address

vote VARCHAR(10) CHECK (vote IN ('yes', 'no', 'abstain')),

weight DOUBLE PRECISION DEFAULT 1.0, -- for weighted voting

timestamp TIMESTAMPTZ DEFAULT NOW(),

blockchain_tx_hash VARCHAR(100) -- reference to on-chain record

);

-- Audit log (synced with blockchain)

CREATE TABLE audit_log (

id BIGSERIAL PRIMARY KEY,

timestamp TIMESTAMPTZ DEFAULT NOW(),

event_type VARCHAR(50) NOT NULL, -- 'metric_calculated', 'policy_created', etc.
```

entity_id VARCHAR(100),

details JSONB,

blockchain_hash VARCHAR(100), -- hash committed to chain

blockchain_block BIGINT

);

-- Indexes for performance

CREATE INDEX idx_metrics_raw_timestamp ON metrics_raw (timestamp DESC);

CREATE INDEX idx_metrics_raw_city_type ON metrics_raw (city_id, metric_type);

CREATE INDEX idx_policies_city_status ON policies (city_id, status);

CREATE INDEX idx_audit_log_timestamp ON audit_log (timestamp DESC);

````

3.3 Layer 3: Policy AI

3.3.1 Policy Orchestrator

````yaml

Purpose: Coordinate multiple AI models for policy prediction

Architecture:

Ensemble of 3+ models:

1. External API (Claude via Anthropic API)

2. External API (Kimi via Moonshot API)

3. Local fine-tuned model (Llama 3 8B OR Mistral 7B)

4. (Optional) External API (Gemini via Google AI API)

Workflow:

1. Receive policy proposal (text + structured parameters)

2. Validate input (schema check, length limits)

3. Parallel invocation of all models

4. Collect predictions (ΔVR, ΔEC, rationale)

5. Aggregate results (see aggregation logic below)

6. Generate explanation (see Explainability Engine)

7. Log to audit trail + blockchain

8. Return to user/dashboard

Aggregation Logic:

- If all models agree (variance < 0.1): Use mean

- If models disagree:

* Flag high uncertainty

* Use confidence-weighted average

* Present dissenting views in explanation

- If any model predicts catastrophic outcome (ΔVR or ΔEC < -0.5):

* Automatic red flag to human review

Implementation:

Service: policy-orchestrator-service (Python FastAPI)

Key Functions:

async def predict_policy_impact(policy: PolicyProposal) -> PolicyPrediction

async def invoke_model(model_id: str, policy: PolicyProposal) -> ModelPrediction

def aggregate_predictions(predictions: List[ModelPrediction]) -> AggregatedPrediction

async def log_to_blockchain(prediction: PolicyPrediction) -> str  # returns tx_hash

API Specification (OpenAPI):

```yaml

POST /v1/policy/predict

Request Body:
```

{

"policy": {

"title": "string",

"description": "string (max 5000 chars)",

"scope": "city|region|global",

"horizon": "integer (years, 1-20)",

"parameters": {

"budget": "float (optional)",

"affected_districts": ["array of strings (optional)"]

}

},

"requester_id": "string (user or system)"

}

Response (200 OK):

{

"prediction_id": "uuid",

"timestamp": "ISO 8601",

"policy": { ... },

"predicted_impact": {

"delta_vr": "float",

"delta_ec": "float",

"delta_vai": "float (null in MVP)",

"delta_sp": "float (null in MVP)"

},

"uncertainty": "float (0-1)",

"confidence": "float (0-1)",

"rationale": {

"summary": "string (1 sentence)",

"detailed": "string (1-2 paragraphs)",

"causal_factors": [

{

"factor": "string",

"impact": "float",

"explanation": "string"

}

]

},

"model_breakdown": [

{

"model_id": "claude-3-opus",

"delta_vr": "float",

"delta_ec": "float",

"confidence": "float"

},

{ ... other models ... }

],

"red_flags": ["array of strings (warnings)"],

"blockchain_tx_hash": "string"

}

Error Responses:

400: Invalid input (schema validation failed)

429: Rate limit exceeded

503: Service unavailable (model API down)

```

---

**Продолжение TDD следует...**

Токенов много использовал. Сейчас продолжу остальные секции:

- 3.3.2 Explainability Engine

- 3.3.3 AI Model Specifications

- 3.4 Layer 4: Governance

- 3.5 Layer 5: Regeneration

- IV. Data Models

- V. API Specs (полные)

- VI. Security

- VII. Deployment

- VIII-XII остальное
```

3.3.2 Explainability Engine

yaml

Purpose: Generate human-understandable explanations for Policy AI predictions

Three-Tier Explanation System:

TIER 1: One-Sentence Summary (Always)

Format: "This policy would {increase|decrease} {metric} by {value}

because {primary_reason}."

Example: "This policy would increase VR by 0.15 because it improves

institutional transparency through open data mandates."

Generation: Template-based with dynamic values

Latency: <100ms

TIER 2: One-Paragraph Rationale (Always)

Format: Structured paragraph with:

- Primary impact explanation

- Secondary effects

- Confidence caveats

- Temporal considerations

Example:

"The urban green space expansion policy would increase EC by 0.22 over 5 years by improving air quality (primary) and enhancing social cohesion through shared public spaces (secondary). However, there's a short-term decrease in VR (-0.05) during construction due to disrupted traffic patterns. Confidence: 0.78 (historical precedents from similar initiatives in Copenhagen and Singapore)."

Generation: LLM-based (Claude/Kimi) with structured prompts

Latency: 2-5 seconds

TIER 3: Full Causal Graph (On-Demand)

Format: Interactive visualization showing:

- Input factors (policy parameters)

- Intermediate variables (mechanisms)

- Metric impacts (VR, EC, VAI, SP)

- Temporal evolution (projections over time)

- Confidence intervals at each node

Technology:

- D3.js for graph visualization

- Causal inference library (DoWhy by Microsoft)

- Historical data for counterfactuals

Generation: Compute-intensive, cached results

Latency: 10-30 seconds (first request), <1s (cached)

Implementation:

Service: explainability-service (Python FastAPI)

Dependencies:

- SHAP 0.42+ (for feature importance)

- Captum (PyTorch attribution)

- DoWhy 0.9+ (causal inference)

- spaCy 3.6+ (NLP for text generation)

Key Functions:

def generate_tier1(prediction: PolicyPrediction) -> str

def generate_tier2(prediction: PolicyPrediction) -> str

def generate_tier3(prediction: PolicyPrediction) -> CausalGraph

def extract_causal_factors(prediction: PolicyPrediction) -> List[Factor]

API Endpoint:

GET /v1/explanations/{prediction_id}?tier=1|2|3

Response:

{

"prediction_id": "uuid",

"tier": "integer (1-3)",

"explanation": {

"summary": "string (tier 1)",

"rationale": "string (tier 2, optional)",

"causal_graph": "object (tier 3, optional)"

},

"generated_at": "ISO 8601",

"model_used": "string (claude-3-opus, etc.)"

}

Caching Strategy:

- Tier 1: No cache (fast enough)

- Tier 2: Redis cache, TTL 24 hours

- Tier 3: Persistent cache in DB, invalidate on model update

Quality Assurance:

- Human review of 10% of explanations (random sample)

- User feedback mechanism ("Was this helpful? Yes/No")

- A/B testing different explanation styles

3.3.3 AI Model Specifications

Model 1: Claude (Anthropic API)

yaml

Model: claude-3-opus-20240229

Purpose: Ethical reasoning, long-term impacts, risk analysis

Integration:

API: Anthropic API (REST)

Authentication: API key (stored in Vault)

Rate Limit: 50 requests/minute

Timeout: 60 seconds

Prompt Template:

System: |

You are an AI advisor for the Symbiotic Codes project. Your role is

to predict the impact of policy proposals on civilizational metrics

(S(s) with components) with focus on:

- Long-term consequences (5-20 years)

- Ethical considerations

- Risk to vulnerable populations

- Biotic diversity (BD) impacts

Current city baseline: VR={vr_current}, EC={ec_current}

Respond in JSON format only.

User: |

Policy Proposal:

Title: {policy_title}

Description: {policy_description}

Scope: {policy_scope}

Horizon: {policy_horizon} years

Predict:

1. delta_vr (change in VR, -5.0 to +5.0)

2. delta_ec (change in EC, -5.0 to +5.0)

3. confidence (0.0 to 1.0)

4. rationale (2-3 sentences)

5. red_flags (array of concerns)

Output JSON:

{

"delta_vr": float,

"delta_ec": float,

"confidence": float,

"rationale": "string",

"red_flags": ["string"]

}

Error Handling:

- Retry 3 times with exponential backoff

- If all retries fail: Mark as "model_unavailable"

- Log error to monitoring system

- Proceed with remaining models

Cost Estimation:

- ~$0.015 per prediction (assuming 1K input + 500 output tokens)

- Monthly budget: $500 (~33K predictions)

Model 2: Kimi (Moonshot AI API)

yaml

Model: kimi-v1 (or latest available)

Purpose: Long-context analysis, multi-source data integration

Integration:

API: Moonshot AI API

Authentication: API key

Rate Limit: (TBD based on API tier)

Timeout: 90 seconds (longer context processing)

Prompt Template:

[Similar structure to Claude, but emphasize:]

- Cross-referencing with historical data

- Semantic coherence analysis

- Temporal fairness considerations

Unique Contribution:

- Analyzes 100K+ tokens of historical policy data

- Identifies similar past policies and their outcomes

- Flags inconsistencies with established norms

Cost: (TBD based on pricing)

Model 3: Local Fine-Tuned Model

yaml

Base Model: Llama 3 8B OR Mistral 7B

Fine-Tuned On:

- Historical policy data (Barcelona + global precedents)

- Synthetic scenarios (generated via GPT-4)

- Human-labeled examples (expert annotations)

Purpose:

- Fast inference (no API latency)

- Cost-effective (no per-request fees)

- Privacy (sensitive data never leaves infrastructure)

Training:

Dataset Size: 10K policy-prediction pairs (minimum)

Training Time: ~48 hours on 4x A100 GPUs

Evaluation:

- Accuracy on held-out test set (target: >75%)

- Correlation with Claude/Kimi predictions (target: r > 0.80)

Deployment:

- TorchServe OR FastAPI + PyTorch

- GPU instance (AWS g5.xlarge OR equivalent)

- Autoscaling: 1-3 replicas based on load

- Model versioning: MLflow

Inference:

Input: JSON (same schema as API models)

Output: JSON prediction

Latency: 500ms - 2s (depending on input length)

Update Cadence:

- Retrain monthly with new data

- A/B test new version vs current

- Gradual rollout (10% → 50% → 100% traffic)

3.3.4 Simulation Sandbox

yaml

Purpose: Allow users to test policy ideas before formal proposal

Features:

- What-if scenarios ("What if we increase green space by 20%?")

- Interactive parameter adjustment (sliders for budget, timeline, etc.)

- Instant prediction (using local model for speed)

- Historical comparison (similar past policies)

- Share results (permalink for discussions)

Implementation:

Frontend: React component in Dashboard

Backend: /v1/policy/simulate endpoint (priority queue)

Limits (Anti-Abuse):

- Anonymous: 5 simulations per day

- Authenticated: 50 simulations per day

- Results cached (identical inputs → return cached)

UI Flow:

1. User enters policy description (text area)

2. Adjusts parameters (sliders: budget, duration, scope)

3. Click "Simulate"

4. Loading animation (~2-5s)

5. Results displayed:

- Predicted ΔVR, ΔEC (gauge charts)

- Tier 1 explanation

- Confidence intervals (error bars)

- Button: "Explore Details" (opens Tier 3 causal graph)

- Button: "Submit as Formal Proposal" (if satisfied)

Educational Value:

- Helps citizens understand how policies affect metrics

- Builds intuition for symbiotic decision-making

- Reduces low-quality proposals (self-filter before submission)

Analytics:

- Track most-simulated policy types

- Identify common citizen concerns

- Feed insights back to Policy AI training

3.4 Layer 4: Governance & Consensus

3.4.1 DAO Interface

yaml

Purpose: Decentralized governance for policy proposals and system changes

MVP Scope:

- Basic proposal creation (text + vote period)

- Simple majority voting (quadratic voting in Phase 2)

- Proposal execution (manual in MVP, automated later)

- Transparency logs (all votes on-chain)

Smart Contracts (Solidity):

GovernanceDAO.sol:

State Variables:

- proposals: mapping(uint256 => Proposal)

- votes: mapping(uint256 => mapping(address => Vote))

- proposalCount: uint256

- votingPeriod: uint256 (default: 7 days)

- executionDelay: uint256 (default: 2 days)

Structs:

Proposal {

id: uint256

proposer: address

title: string

description: string

ipfsHash: string (full details on IPFS)

createdAt: uint256

votingDeadline: uint256

votesFor: uint256

votesAgainst: uint256

votesAbstain: uint256

executed: bool

status: ProposalStatus (Pending|Active|Passed|Failed|Executed)

}

Vote {

voter: address

choice: VoteChoice (For|Against|Abstain)

weight: uint256 (1 in MVP, variable in Phase 2)

timestamp: uint256

}

Functions:

function createProposal(string title, string description, string ipfsHash)

public returns (uint256 proposalId)

function vote(uint256 proposalId, VoteChoice choice) public

function executeProposal(uint256 proposalId) public

// Only after voting period + execution delay

// Requires proposal to have passed (votesFor > votesAgainst)

function getProposal(uint256 proposalId) public view returns (Proposal)

function getVote(uint256 proposalId, address voter) public view returns (Vote)

Events:

event ProposalCreated(uint256 indexed proposalId, address indexed proposer)

event VoteCast(uint256 indexed proposalId, address indexed voter, VoteChoice choice)

event ProposalExecuted(uint256 indexed proposalId)

AuditTrail.sol:

Purpose: Store hashes of critical events (metrics, predictions, votes)

Function:

function logEvent(string eventType, bytes32 dataHash) public

// Only callable by authorized contracts

// Emits LogEntry event with timestamp

Usage:

- Metric calculations → commit hash daily

- Policy predictions → commit hash per prediction

- Governance votes → commit hash per vote

Frontend (React):

Pages:

/governance

- List all proposals (filterable by status)

- Create new proposal button

/governance/proposals/:id

- Proposal details

- Vote buttons (For / Against / Abstain)

- Vote distribution chart

- Comments section (off-chain, in DB)

/governance/create

- Form: Title, Description, Upload supporting docs

- Preview before submission

- Submit → Creates on-chain proposal + uploads to IPFS

Web3 Integration:

- MetaMask connection (or WalletConnect for mobile)

- Transaction signing for votes

- Gas fee estimation before actions

- Transaction status tracking

Access Control (MVP):

- Anyone can view proposals

- Only verified citizens can vote (KYC via pilot city)

- Only TSC members can execute proposals (manual in MVP)

3.4.2 Human Review Interface

yaml

Purpose: Human-in-loop for critical decisions

Use Cases:

1. High-uncertainty predictions (confidence < 0.6)

2. Catastrophic risk flags (ΔVR or ΔEC < -0.5)

3. Policy proposals affecting >10K citizens

4. System parameter changes (voting thresholds, etc.)

Workflow:

1. Trigger: Automated flag OR manual escalation

2. Notification: Email + Dashboard alert to reviewers

3. Review: Examine prediction, read rationale, check data sources

4. Decision: Approve / Reject / Request More Info

5. Documentation: Record rationale for decision

6. Log: All actions logged to audit trail + blockchain

Reviewers (MVP):

- Technical Steering Committee (TSC): 5 members

- City Officials: 2 designated liaisons

- Citizen Representatives: 3 rotating (sortition, quarterly)

Total: 10 reviewers

Quorum: 6/10 for decisions

Interface:

Admin Dashboard (/admin/review):

- Pending reviews queue (sorted by urgency)

- Each item shows:

* Policy summary

* AI prediction + explanation

* Red flags (highlighted)

* Data quality indicators

* Historical precedents

- Actions:

* Approve (with optional notes)

* Reject (must provide reason)

* Request additional analysis (specify what's needed)

* Escalate to full TSC meeting

Accountability:

- All decisions attributed to reviewer (pseudonymous in public logs)

- Periodic audits of review quality

- Override mechanism if community challenges decision (via DAO vote)

SLA (Service Level Agreement):

- High urgency: Review within 24 hours

- Medium: 72 hours

- Low: 7 days

- Overdue items → automatic escalation

3.5 Layer 5: Regeneration & Future Weighting

3.5.1 Biotic Diversity (BD) Monitor

yaml

Purpose: Ensure policies don't degrade ecological health

Metric Definition:

BD = biological_diversity_index × ecosystem_health_score

Target: BD ≥ 0.25 (from Vector I)

Warning Threshold: BD < 0.30

Circuit Breaker: BD < 0.25 for 2+ consecutive quarters

Data Sources:

1. Satellite imagery (NDVI - Normalized Difference Vegetation Index)

2. Biodiversity databases (iNaturalist, GBIF citizen science data)

3. City environmental reports (species counts, habitat assessments)

4. Air/Water quality sensors (proxy for ecosystem health)

Calculation (Simplified for MVP):

biological_diversity_index = species_richness / reference_baseline

ecosystem_health_score = weighted_avg([

air_quality_score,

water_quality_score,

green_space_ratio,

habitat_connectivity

])

Implementation:

Service: bd-monitor-service (Python)

Frequency: Weekly calculations, monthly reports

Alerting:

- If BD < 0.30: Yellow alert to dashboard

- If BD < 0.25: Red alert + automatic escalation to TSC

- If policy prediction shows ΔBD < -0.05: Flag in Policy AI

Integration with Policy AI:

- Policy AI must include BD impact in predictions

- Policies with negative BD impact require higher approval threshold

- Alternative policies suggested if BD risk detected

Visualization (Dashboard):

- BD trend over time (line chart)

- Current BD score (gauge with green/yellow/red zones)

- Species diversity heatmap (by city district)

- Policy impact on BD (for each proposal)

3.5.2 Future Agent (Temporal Weighting)

yaml

Purpose: Represent interests of future generations (2050-2100)

Conceptual Role:

- "Virtual stakeholder" in governance decisions

- Applies long-term discount factor to short-term gains

- Flags policies that benefit present at expense of future

Implementation (MVP - Simplified):

Rule-Based System (Phase 2: ML-based):

Rules:

1. Temporal Discount:

- Short-term gains (<5 years) weighted × 0.7

- Long-term impacts (>10 years) weighted × 1.5

2. Sustainability Check:

- If policy depletes non-renewable resource → Flag

- If policy creates long-term debt (financial or ecological) → Flag

3. Generational Equity:

- Calculate benefit distribution across age cohorts

- If >70% benefits accrue to current generation → Flag

4. Irreversibility Risk:

- Identify irreversible consequences (e.g., species extinction)

- Assign high penalty in scoring

Workflow:

1. Policy AI generates prediction

2. Future Agent applies temporal weighting

3. If long-term score < short-term score × 0.8:

→ Flag as "Future Risk"

4. Add Future Agent perspective to explanation

5. Require explicit justification from proposer

Output (Added to Policy Prediction):

{

"future_agent_assessment": {

"temporal_weighted_score": float,

"short_term_bias_detected": boolean,

"long_term_risks": ["array of strings"],

"recommendation": "proceed|caution|redesign"

}

}

Example:

Policy: "Expand highway to reduce traffic congestion"

Short-term (5 years):

- ΔVR: +0.2 (improved mobility)

- ΔEC: -0.1 (construction impact)

Long-term (20 years):

- ΔVR: -0.1 (induced demand, more congestion)

- ΔEC: -0.3 (carbon lock-in, sprawl)

- ΔBD: -0.15 (habitat fragmentation)

Future Agent Flag:

"This policy prioritizes short-term convenience over long-term

sustainability. Consider alternatives: public transit expansion,

congestion pricing, urban densification."

Governance Integration:

- Future Agent assessment shown in all policy proposals

- TSC must explicitly override Future Agent concerns if proceeding

- Annual report: "Policies overridden despite Future Agent warnings"

## IV. DATA MODELS & SCHEMAS

4.1 Core Data Types

typescript

```
// TypeScript definitions for type safety across frontend/backend

// ============== Metrics ==============

interface MetricValue {

value: number;           // The metric score

confidence: number;      // 0.0 - 1.0

timestamp: string;       // ISO 8601

source: string;          // Data source identifier

metadata?: Record<string, any>;

}

interface VRMetric extends MetricValue {

metric_type: 'VR';

breakdown: {

institutional_transparency: number;

decision_making_speed: number;

policy_effectiveness: number;
```

```
citizen_engagement: number;

};

}

interface ECMetric extends MetricValue {

metric_type: 'EC';

breakdown: {

bio_health: number;

tech_integration: number;

cultural_coherence: number;

};

}

// ============== Policies ==============

interface PolicyProposal {

id?: string;              // UUID, generated on creation

city_id: string;

title: string;
```

```
description: string;     // Max 5000 chars

scope: 'city' | 'region' | 'global';

horizon: number;         // Years (1-20)

parameters?: {

budget?: number;

affected_districts?: string[];

[key: string]: any;

};

created_by: string;      // User ID or system

created_at?: string;     // ISO 8601

status: 'draft' | 'proposed' | 'active' | 'completed' | 'rejected';

}

interface PolicyPrediction {

prediction_id: string;   // UUID

policy: PolicyProposal;

timestamp: string;

predicted_impact: {
```

```
delta_vr: number;

delta_ec: number;

delta_vai: number | null;  // Null in MVP

delta_sp: number | null;   // Null in MVP

};

uncertainty: number;     // 0.0 - 1.0

confidence: number;      // 0.0 - 1.0

rationale: {

summary: string;       // Tier 1 explanation

detailed: string;      // Tier 2 explanation

causal_factors: CausalFactor[];

};

model_breakdown: ModelPrediction[];

red_flags: string[];

future_agent_assessment?: FutureAgentAssessment;

blockchain_tx_hash?: string;
```

```
}

interface ModelPrediction {

model_id: string;        // 'claude-3-opus', 'kimi-v1', 'local-llama3'

delta_vr: number;

delta_ec: number;

confidence: number;

rationale?: string;

}

interface CausalFactor {

factor: string;          // E.g., "Increased green space"

impact: number;          // Magnitude of impact on metrics

explanation: string;

}

interface FutureAgentAssessment {

temporal_weighted_score: number;

short_term_bias_detected: boolean;

long_term_risks: string[];
```

```
recommendation: 'proceed' | 'caution' | 'redesign';

}

// ============== Governance ==============

interface Proposal {

id: number;              // On-chain ID

proposer: string;        // Ethereum address

title: string;

description: string;

ipfs_hash: string;       // Full details on IPFS

created_at: number;      // Unix timestamp

voting_deadline: number;

votes_for: number;

votes_against: number;

votes_abstain: number;

executed: boolean;

status: 'Pending' | 'Active' | 'Passed' | 'Failed' | 'Executed';
```

```
}

interface Vote {

voter: string;           // Ethereum address

choice: 'For' | 'Against' | 'Abstain';

weight: number;          // 1 in MVP

timestamp: number;

blockchain_tx_hash: string;

}

// ============== Users ==============

interface User {

id: string;              // UUID

wallet_address?: string; // Ethereum address (if connected)

city_id: string;

role: 'citizen' | 'official' | 'tsc_member' | 'admin';

verified: boolean;       // KYC status

opt_in_level: 'basic' | 'full' | 'none';  // Data sharing consent

created_at: string;
```

```
last_active: string;

}

// ============== Audit Logs ==============

interface AuditLogEntry {

id: number;

timestamp: string;       // ISO 8601

event_type: string;      // 'metric_calculated', 'policy_created', etc.

entity_id: string;

details: Record<string, any>;

blockchain_hash: string;

blockchain_block: number;

}
```

4.2 API Request/Response Schemas (JSON Schema)

```
json

// POST /v1/policy/predict Request Schema

{
```

```
"$schema": "http://json-schema.org/draft-07/schema#",

"type": "object",

"required": ["policy"],

"properties": {

"policy": {

"type": "object",

"required": ["title", "description", "scope", "horizon"],

"properties": {

"title": {

"type": "string",

"minLength": 10,

"maxLength": 200

},

"description": {

"type": "string",

"minLength": 50,

"maxLength": 5000
```

},

"scope": {

"type": "string",

"enum": ["city", "region", "global"]

},

"horizon": {

"type": "integer",

"minimum": 1,

"maximum": 20

},

"parameters": {

"type": "object",

"additionalProperties": true

}

}

},

"requester_id": {

"type": "string",

"format": "uuid"

}

}

}

// Response Schema (200 OK)

{

"$schema": "http://json-schema.org/draft-07/schema#",

"type": "object",

"required": ["prediction_id", "timestamp", "policy", "predicted_impact", "uncertainty", "confidence", "rationale"],

"properties": {

"prediction_id": {

"type": "string",

"format": "uuid"

},

"timestamp": {

"type": "string",

"format": "date-time"

},

"policy": {

"$ref": "#/definitions/PolicyProposal"

},

"predicted_impact": {

"type": "object",

"properties": {

"delta_vr": {"type": "number"},

"delta_ec": {"type": "number"},

"delta_vai": {"type": ["number", "null"]},

"delta_sp": {"type": ["number", "null"]}

}

},

"uncertainty": {

"type": "number",

"minimum": 0,

"maximum": 1

},

"confidence": {

"type": "number",

"minimum": 0,

"maximum": 1

},

"rationale": {

"type": "object",

"properties": {

"summary": {"type": "string"},

"detailed": {"type": "string"},

"causal_factors": {

"type": "array",

"items": {

```
"type": "object",

"properties": {

"factor": {"type": "string"},

"impact": {"type": "number"},

"explanation": {"type": "string"}

}

}

}

}

},

"model_breakdown": {

"type": "array",

"items": {

"$ref": "#/definitions/ModelPrediction"

}

},
```

"red_flags": {

"type": "array",

"items": {"type": "string"}

},

"blockchain_tx_hash": {"type": "string"}

}

}

V. API SPECIFICATIONS (Complete)

5.1 API Overview

yaml

Base URL (Production): https://api.symbiotic.city

Base URL (Staging): https://api-staging.symbiotic.city

Base URL (Local Dev): http://localhost:8000

Versioning: /v1/ prefix (v2, v3 in future)

Authentication: JWT tokens (short-lived) + API keys (for services)

Rate Limiting:

- Public endpoints: 100 req/hour per IP

- Authenticated: 1000 req/hour per user

- Whitelisted (partners): 10000 req/hour

Response Format: JSON

Timestamps: ISO 8601 (UTC)

Error Format: RFC 7807 (Problem Details)

5.2 Endpoint Catalog

yaml

# ============= Public Endpoints =============

GET /v1/health

Description: Service health check

Auth: None

Response: { "status": "ok", "timestamp": "..." }

GET /v1/metrics/{city_id}

Description: Get latest metric scores for a city

Auth: None (public data)

Params:

- city_id: string (path)

- metrics: string[] (query, optional, default: all)

Example: ?metrics=VR,EC

Response:

{

"city_id": "barcelona",

"timestamp": "2025-10-21T12:00:00Z",

"metrics": {

## "VR": {

"value": 2.35,

"confidence": 0.82,

"breakdown": {...}

},

## "EC": {

"value": 1.78,

"confidence": 0.75,

"breakdown": {...}

}

}

}

GET /v1/metrics/{city_id}/history

Description: Historical metric data

Auth: None

Params:

- city_id: string (path)

- metric: string (query, required, e.g., "VR")

- start: ISO 8601 date (query)

- end: ISO 8601 date (query)

- granularity: enum (hourly|daily|weekly|monthly)

Response:

{

"city_id": "barcelona",

"metric": "VR",

"granularity": "daily",

"data": [

{

"timestamp": "2025-10-01T00:00:00Z",

"value": 2.30,

"confidence": 0.80

},

...

]

}

GET /v1/governance/proposals

Description: List all governance proposals

Auth: None

Params:

- status: enum (query, optional): Pending|Active|Passed|Failed|Executed

- page: integer (query, default: 1)

- limit: integer (query, default: 20, max: 100)

Response:

{

"proposals": [

{

"id": 42,

"title": "Increase bike lane funding",

"status": "Active",

"created_at": "...",

"voting_deadline": "...",

"votes_for": 1523,

"votes_against": 342,

"votes_abstain": 87

},

...

],

"pagination": {

"page": 1,

"limit": 20,

"total": 156

}

}

GET /v1/governance/proposals/{proposal_id}

Description: Get proposal details

Auth: None

Response:

{

"proposal": {...},  // Full Proposal object

"votes": {

"distribution": {...},  // Breakdown by district, demographics

"recent": [...]  // Last 10 votes (anonymized)

}

}

POST /v1/auth/login

Description: Authenticate user and get JWT token

Auth: None (this is the login endpoint)

Body:

{

"wallet_address": "string (Ethereum address, optional)",

"email": "string (optional)",

"verification_code": "string (sent via email/SMS)"

}

Response:

{

"access_token": "JWT string",

"token_type": "Bearer",

"expires_in": 3600,  // seconds

"user": {

"id": "uuid",

"city_id": "barcelona",

"role": "citizen",

"verified": true

}

}

Errors:

401: Invalid credentials

429: Too many attempts

POST /v1/auth/refresh

Description: Refresh expired JWT token

Auth: Refresh token (in cookie or header)

Response: New access_token

POST /v1/data/ingest

Description: Submit data from apps or IoT devices

Auth: API key (for IoT) OR JWT (for apps)

Body:

{

"source_id": "string",

"timestamp": "ISO 8601",

"metric_type": "enum (air_quality|traffic|survey_response|...)",

"value": "float or object",

"location": {

"lat": "float",

"lon": "float"

},

"metadata": "object (optional)"

}

Response:

{

"status": "accepted",

"ingestion_id": "uuid",

"processed_at": "ISO 8601"

}

Errors:

400: Invalid schema

401: Unauthorized

422: Data validation failed

429: Rate limit exceeded

GET /v1/user/profile

Description: Get current user profile

Auth: JWT required

Response:

{

"user": {

"id": "uuid",

"wallet_address": "string (optional)",

"city_id": "barcelona",

"role": "citizen",

"verified": true,

"opt_in_level": "full",

"created_at": "ISO 8601",

"data_contributions": {

"surveys_completed": 42,

"last_contribution": "ISO 8601"

}

}

}

PUT /v1/user/profile

Description: Update user preferences

Auth: JWT required

Body:

{

"opt_in_level": "basic|full|none",

"notification_preferences": {...}

}

Response: Updated user object

POST /v1/policy/predict

Description: Request policy impact prediction

Auth: JWT required

Body: (as specified in Section 3.3.1)

Response: (as specified in Section 3.3.1)

Rate Limit: 10 predictions/hour per user

Cost: Free for verified citizens (MVP)

POST /v1/policy/simulate

Description: Run what-if simulation (non-persistent)

Auth: Optional (anonymous limited to 5/day)

Body: Similar to /predict but marked as simulation

Response: Prediction without blockchain logging

Rate Limit:

- Anonymous: 5/day

- Authenticated: 50/day

GET /v1/policy/predictions/{prediction_id}

Description: Retrieve past prediction

Auth: JWT required (or public if prediction marked public)

Response: Full PolicyPrediction object

POST /v1/governance/proposals

Description: Create new governance proposal

Auth: JWT required + verified citizen

Body:

{

"title": "string (10-200 chars)",

"description": "string (50-5000 chars)",

"ipfs_hash": "string (optional, for full details)",

"voting_period_days": "integer (default: 7)"

}

Response:

{

"proposal_id": "integer (on-chain ID)",

"blockchain_tx_hash": "string",

"status": "Pending"

}

Errors:

403: User not verified

400: Invalid input

POST /v1/governance/proposals/{proposal_id}/vote

Description: Cast vote on proposal

Auth: JWT required + verified citizen

Body:

{

"choice": "For|Against|Abstain"

}

Response:

{

"vote_id": "uuid",

"blockchain_tx_hash": "string",

"weight": 1.0,

"timestamp": "ISO 8601"

}

Errors:

403: Not eligible to vote

409: Already voted

410: Voting period ended

GET /v1/explanations/{prediction_id}

Description: Get explanation for policy prediction

Auth: None (if prediction is public)

Params:

- tier: integer (1|2|3, default: 1)

Response: (as specified in Section 3.3.2)

# ============= Admin/TSC Endpoints =============

GET /v1/admin/review/queue

Description: Get pending items for human review

Auth: JWT + role:tsc_member OR role:admin

Response:

```
{
"pending_reviews": [
{
"review_id": "uuid",
"type": "policy_prediction|system_change|high_risk",
"priority": "high|medium|low",
"created_at": "ISO 8601",
"deadline": "ISO 8601",
"subject": {...}  // Policy or system change details
},
...
],
"counts": {
"high_priority": 3,
"medium_priority": 7,
"low_priority": 12
```

}

}

POST /v1/admin/review/{review_id}/decision

Description: Submit review decision

Auth: JWT + role:tsc_member OR role:admin

Body:

{

"decision": "approve|reject|request_more_info",

"rationale": "string (required)",

"additional_notes": "string (optional)"

}

Response:

{

"review_id": "uuid",

"decision": "approve",

"reviewer_id": "uuid",

"timestamp": "ISO 8601",

"blockchain_tx_hash": "string"

}

GET /v1/admin/metrics/health

Description: System health dashboard

Auth: JWT + role:admin

Response:

{

"system_status": "healthy|degraded|critical",

"services": {

"api_gateway": {

"status": "healthy",

"uptime": 0.9987,

"response_time_p95": 245  // ms

},

"measurement_service": {...},

"policy_service": {...},

"database": {...},

"blockchain": {...}

},

"alerts": [

{

"severity": "warning",

"message": "Federated learning convergence slow",

"timestamp": "ISO 8601"

}

]

}

POST /v1/admin/system/parameter

Description: Update system parameter (requires TSC vote)

Auth: JWT + role:admin + governance_approval_hash

Body:

{

"parameter_name": "string (e.g., 'voting_period_days')",

"new_value": "any",

"governance_approval_hash": "string (blockchain tx proving vote)"

}

Response:

{

"parameter_name": "voting_period_days",

"old_value": 7,

"new_value": 10,

"applied_at": "ISO 8601",

"blockchain_tx_hash": "string"

}

# ============= Internal/Service-to-Service Endpoints =============

POST /internal/v1/privacy/process

Description: Apply privacy layer to raw data

Auth: Service API key (internal only)

Body:

{

"data": [...],  // Array of raw data points

"privacy_config": {

"epsilon": 0.5,

"noise_type": "Laplace"

}

}

Response:

{

"processed_data": [...],  // Anonymized data

"privacy_guarantee": "ε=0.5 differential privacy applied"

}

POST /internal/v1/federated/aggregate

Description: Aggregate federated learning updates

Auth: Service API key

Body:

{

"round_id": "integer",

"client_updates": [

{

"client_id": "string",

"gradients": [...],  // Model gradients

"num_samples": "integer"

},

...

]

}

Response:

{

"global_model": {...},  // Updated model weights

"convergence_metric": 0.0023,

"next_round_id": "integer"

}

POST /internal/v1/blockchain/commit

Description: Commit hash to blockchain

Auth: Service API key

Body:

{

"event_type": "string",

"data_hash": "string (SHA-256)",

"metadata": "object (optional)"

}

Response:

{

"blockchain_tx_hash": "string",

"block_number": "integer",

"gas_used": "integer",

"timestamp": "ISO 8601"

}

5.3 Error Handling Standard

yaml

Error Response Format (RFC 7807):

{

"type": "string (URI identifying error type)",

"title": "string (human-readable summary)",

"status": "integer (HTTP status code)",

"detail": "string (detailed explanation)",

"instance": "string (URI of specific occurrence)",

"timestamp": "ISO 8601",

"request_id": "string (for tracing)"

}

Example:

{

"type": "https://api.symbiotic.city/errors/rate-limit-exceeded",

"title": "Rate Limit Exceeded",

"status": 429,

"detail": "You have exceeded 100 requests per hour. Limit resets at 2025-10-21T14:00:00Z.",

"instance": "/v1/policy/predict",

"timestamp": "2025-10-21T13:42:17Z",

"request_id": "req_a7b3d9f2"

}

Standard Error Codes:

400 Bad Request:

- Invalid JSON syntax

- Schema validation failed

- Missing required fields

401 Unauthorized:

- Missing authentication token

- Invalid or expired token

403 Forbidden:

- User lacks required role/permission

- Action not allowed in current state

404 Not Found:

- Resource doesn't exist

- Endpoint not found

409 Conflict:

- Resource already exists

- State conflict (e.g., already voted)

422 Unprocessable Entity:

- Data validation failed (business logic)

- Constraint violation

429 Too Many Requests:

- Rate limit exceeded

- Include Retry-After header

500 Internal Server Error:

- Unexpected server error

- Log for investigation

503 Service Unavailable:

- Dependency unavailable (DB, AI API)

- Temporary maintenance

Retry Strategy (for clients):

- 429: Wait for Retry-After duration

- 500/503: Exponential backoff (1s, 2s, 4s, 8s, 16s max)

- 4xx (except 429): Don't retry, fix request

5.4 Pagination Standard

yaml

Query Parameters:

page: integer (1-indexed, default: 1)

limit: integer (default: 20, max: 100)

sort: string (field name, e.g., "created_at")

order: "asc" OR "desc" (default: "desc")

Response Format:

{

"data": [...],  // Array of items

"pagination": {

"page": 1,

"limit": 20,

"total_items": 156,

"total_pages": 8,

"has_next": true,

"has_prev": false,

"next_page": "/v1/resource?page=2&limit=20",

"prev_page": null

}

}

Link Headers (RFC 5988):

Link: <https://api.symbiotic.city/v1/resource?page=2&limit=20>; rel="next",

<https://api.symbiotic.city/v1/resource?page=8&limit=20>; rel="last"

5.5 Versioning Strategy

yaml

Approach: URL Path Versioning (/v1/, /v2/, etc.)

Version Lifecycle:

- v1 (MVP): Current, stable

- v2 (Future): 6-month deprecation notice before v1 sunset

Backward Compatibility:

- Additive changes: New optional fields → no version bump

- Breaking changes: Require new version

Breaking Changes:

- Removing fields

- Changing field types

- Changing error codes

- Changing authentication method

Sunset Policy:

1. Announce deprecation (6 months notice)

2. Add Sunset header: Sunset: Sat, 01 May 2026 00:00:00 GMT

3. Add Deprecation header: Deprecation: true

4. Gradually reduce rate limits on old version

5. Remove old version after grace period

## VI. SECURITY & PRIVACY

6.1 Authentication & Authorization

yaml

Authentication Methods:

1. JWT (JSON Web Tokens):

Purpose: User authentication (mobile app, web dashboard)

Token Structure:

Header: {"alg": "RS256", "typ": "JWT"}

Payload: {

"sub": "user_uuid",

"city_id": "barcelona",

"role": "citizen",

"verified": true,

"iat": 1698765432,  // issued at

"exp": 1698769032   // expires (1 hour)

}

Signature: RSA256(header + payload, private_key)

Issuance:

- POST /v1/auth/login

- Verify user credentials (email+code OR wallet signature)

- Generate access token (1 hour) + refresh token (30 days)

Refresh:

- POST /v1/auth/refresh with refresh token

- Issue new access token

- Rotate refresh token every 7 days

Revocation:

- Store revoked tokens in Redis (TTL = expiry time)

- Check blacklist on each request

- User logout → revoke all tokens for that user

2. API Keys:

Purpose: Service-to-service authentication (IoT devices, internal services)

Format: symc_live_a7b3d9f2e8c1...  (32-char random)

Storage:

- Hashed (SHA-256) in database

- Never stored in plaintext

- Display only once at creation

Rotation:

- Quarterly for production keys

- Alert 30 days before expiry

- Graceful migration (both old and new valid during transition)

Scopes:

- data:ingest (IoT devices)

- metrics:read (public dashboards)

- internal:all (service-to-service)

3. Wallet Signatures (Optional):

Purpose: Ethereum wallet authentication for governance

Flow:

1. User connects wallet (MetaMask)

2. Backend generates nonce: "Sign this message to authenticate: {nonce}"

3. User signs with private key

4. Backend verifies signature with wallet address

5. Issue JWT if valid

Benefits:

- No password needed

- Cryptographic proof of identity

- Integrates with on-chain governance

Authorization (RBAC):

Roles:

- citizen: Basic user, can submit data, vote

- verified_citizen: KYC completed, full governance rights

- official: City employee, can access sensitive dashboards

- tsc_member: Technical Steering Committee, admin review access

- admin: Full system access

Permissions (Examples):

- data:ingest → citizen, official, admin

- policy:create → verified_citizen, official, admin

- governance:vote → verified_citizen

- admin:review → tsc_member, admin

- system:configure → admin (with governance approval)

Implementation:

- Permissions stored in JWT claims

- Checked via decorators (FastAPI dependencies)

- Cached in Redis for performance

Middleware Stack:

1. Rate Limiting (check before auth)

2. Authentication (validate JWT/API key)

3. Authorization (check permissions)

4. Request Validation (schema check)

5. Business Logic

6. Audit Logging

7. Response

6.2 Data Encryption

yaml

Encryption at Rest:

Database (TimescaleDB):

- Full disk encryption (LUKS or cloud provider native)

- Encrypted backups (AES-256)

- Column-level encryption for sensitive fields:

* User emails: encrypted with application key

* Wallet addresses: hashed (one-way)

Key Management:

- Master key in HashiCorp Vault

- Automatic key rotation (yearly)

- Separate keys per environment (prod/staging/dev)

Files (IPFS):

- Public files: no encryption (immutable, auditable)

- Private files: AES-256 before upload, key stored in Vault

Secrets (API keys, passwords):

- Never in code or config files

- Stored in HashiCorp Vault

- Injected at runtime via environment variables

- Accessed via Vault API with short-lived tokens

Encryption in Transit:

TLS Configuration:

- TLS 1.3 only (no TLS 1.2 or below)

- Strong cipher suites:

## * TLS_AES_256_GCM_SHA384

## * TLS_CHACHA20_POLY1305_SHA256

- Perfect Forward Secrecy (ECDHE)

- HSTS header: max-age=31536000; includeSubDomains; preload

Certificate Management:

- Let's Encrypt (free, auto-renewal)

- Wildcard cert for *.symbiotic.city

- Transparency logs (Certificate Transparency)

Internal Communication:

- Service-to-service: mTLS (mutual TLS)

- Each service has own certificate

- Certificate rotation every 90 days (automated)

Blockchain:

- Private keys stored in hardware security module (HSM) in production

- Multi-signature wallet for critical operations (3-of-5)

- Never expose private keys to application code

6.3 Privacy-Preserving Techniques

yaml

1. Differential Privacy:

Library: OpenDP (Python)

Configuration:

epsilon: 0.5 (privacy budget)

delta: 1e-5 (failure probability)

noise_mechanism: "Laplace" for continuous data, "Geometric" for counts

Application:

- Survey responses: Add noise before aggregation

- Location data: Snap to grid (k-anonymity ≥ 5)

- Time series: Smooth with moving average + noise

Budget Allocation:

- Total ε per user per month: 2.0

- Each query consumes portion of budget

- Alert user when budget low

- Reset monthly

Validation:

- Formal proof that ε-DP is satisfied

- Audited by privacy researcher (annual)

2. Federated Learning:

Framework: Flower

Architecture:

- Server: Coordinates training rounds

- Clients: Mobile apps, IoT devices (10-100 per round)

- No raw data leaves client

Security:

- Secure aggregation (encrypt gradients)

- Byzantine-robust aggregation (detect poisoning)

- Dropout tolerance (train with incomplete clients)

Privacy Guarantees:

- Local Differential Privacy on gradients

- Gradient clipping (prevent large updates)

- Noise addition (Gaussian mechanism)

3. Zero-Knowledge Proofs:

Use Cases:

- Verify user is eligible voter without revealing identity

- Prove metric calculation correct without revealing raw data

Library: ZoKrates (for simple circuits)

Example: Voting Eligibility

Circuit: "Prove age ≥ 18 AND residency in Barcelona"

Public Input: Vote choice

Private Input: Birth date, address

Proof: zk-SNARK that age/residency constraints satisfied

Performance:

- Proof generation: ~1-5 seconds (client-side)

- Verification: <100ms (on-chain or server)

Limitations (MVP):

- Complex circuits expensive

- Use only for high-value scenarios (governance votes)

4. K-Anonymity:

Application: Demographic data in dashboards

Rules:

- Group size minimum: k=5

- If district has <5 residents with attribute X, merge with adjacent districts

- Suppress outlier values

Example:

Original: "District A, Age 23, Income $85K" → Unique

Anonymized: "Districts A-B-C, Age 20-29, Income $80-90K" → k=12

5. Data Minimization:

Principle: Collect only what's necessary

Practices:

- Survey: 5 questions max (not 50)

- Location: District-level (not GPS coordinates)

- Retention: 30 days raw data (not indefinite)

- Access: Role-based (not open to all staff)

User Control:

- Opt-in levels: None / Basic / Full

- Granular permissions (opt-in to surveys but not location)

- Delete account → purge all personal data within 30 days (GDPR)

6.4 Security Protocols

yaml

Secure Development Lifecycle:

1. Code Review:

- All PRs require 2 approvals

- Security-critical PRs require 1 TSC member approval

- Automated checks (linting, type checking)

2. Static Analysis:

Tools:

- Bandit (Python security linter)

- Semgrep (custom security rules)

- npm audit (JavaScript dependencies)

CI/CD Integration:

- Run on every commit

- Block merge if critical issues found

3. Dependency Scanning:

Tools:

- Snyk (vulnerabilities in dependencies)

- Dependabot (automated updates)

Policy:

- High/Critical vulnerabilities: Fix within 7 days

- Medium: 30 days

- Low: Best effort

4. Secret Scanning:

- GitHub secret scanning (automatic)

- TruffleHog (pre-commit hook)

- Never commit API keys, passwords, private keys

5. Penetration Testing:

Frequency: Quarterly (MVP), Monthly (Production)

Scope:

- Web application (OWASP Top 10)

- API endpoints (injection, auth bypass)

- Smart contracts (reentrancy, overflow)

Process:

- Hire external security firm

- Provide findings report

- Track remediation in GitHub issues

- Re-test after fixes

6. Bug Bounty Program:

Launch: Phase 2 (after MVP stabilizes)

Platform: HackerOne or Immunefi

Rewards:

- Critical: $5000 - $10000

- High: $1000 - $5000

- Medium: $500 - $1000

- Low: $100 - $500

Scope:

- Web application and APIs

- Smart contracts

- Infrastructure (if applicable)

Out of Scope:

- Social engineering

- DDoS attacks

- Third-party services

Incident Response Plan:

1. Detection:

- Monitoring alerts (Prometheus, Sentry)

- User reports (security@symbiotic.city)

- Bug bounty submissions

2. Triage (Within 1 hour):

- Assess severity (P0: Critical, P1: High, P2: Medium, P3: Low)

- Assign incident commander

- Create incident channel (Slack)

3. Containment (Within 4 hours for P0/P1):

- Isolate affected systems

- Rotate compromised credentials

- Block malicious IPs

- Deploy emergency patch if possible

4. Investigation:

- Collect logs (preserve for forensics)

- Identify root cause

- Determine scope (what data accessed?)

5. Remediation:

- Fix vulnerability

- Deploy patch to production

- Verify fix with testing

6. Communication:

- Internal: Update stakeholders (TSC, team)

- External: If user data affected, notify within 72 hours (GDPR)

- Public: Post-mortem blog post (after resolution)

7. Post-Mortem (Within 7 days):

- Document timeline

- Root cause analysis (5 Whys)

- Action items to prevent recurrence

- Update runbooks

Security Checklist (Pre-Deployment):

□ All secrets in Vault (not in code/env files)

☐ TLS configured (A+ rating on SSL Labs)

☐ Rate limiting enabled

☐ CORS properly configured

☐ Authentication required on sensitive endpoints

☐ Authorization checks on all actions

☐ Input validation (schema + sanitization)

☐ SQL injection prevention (parameterized queries)

☐ XSS prevention (output encoding)

☐ CSRF tokens on state-changing requests

☐ Audit logging enabled

☐ Monitoring and alerting configured

☐ Backups tested (restore drill)

☐ Incident response plan documented

☐ Security contacts published

## VII. DEPLOYMENT ARCHITECTURE

7.1 Infrastructure Overview

yaml

Environment Strategy:

Development (Local):

- Docker Compose

- Developers run locally

- Connects to testnet (Polygon Mumbai)

Staging:

- Kubernetes cluster (single region)

- Mirrors production architecture

- Uses staging database (copy of prod)

- Connects to testnet

Production:

- Kubernetes cluster (multi-region in Phase 2)

- High availability (3+ replicas per service)

- Connects to mainnet (Polygon)

- Auto-scaling enabled

Cloud Provider:

Primary Choice: AWS (EKS) OR Google Cloud (GKE)

Rationale:

- Mature Kubernetes offerings

- Global infrastructure

- Compliance certifications (SOC 2, ISO 27001)

Alternative: Azure (AKS) if pilot city requires

Multi-Cloud Strategy (Phase 2):

- Terraform for cloud-agnostic provisioning

- Avoid vendor lock-in

- Disaster recovery across clouds

Regions (Production):

MVP: Single region (EU-West-1 for Barcelona)

Phase 2: Multi-region (EU, US, Asia)

Data Residency:

- EU citizen data stays in EU (GDPR)

- Cross-region replication for disaster recovery

7.2 Kubernetes Architecture

yaml

Cluster Configuration:

Node Pools:

1. System Pool (for K8s system components):

- Size: 2 nodes

- Instance: t3.medium (AWS) or n1-standard-2 (GCP)

- Auto-scaling: No (stable)

2. Application Pool (for services):

- Size: 3-10 nodes

- Instance: t3.large or n1-standard-4

- Auto-scaling: Yes (CPU > 70% or memory > 80%)

3. GPU Pool (for ML inference):

- Size: 1-3 nodes

- Instance: g4dn.xlarge (AWS) or n1-highmem-8 + T4 (GCP)

- Auto-scaling: Yes (queue depth)

Namespaces:

- kube-system: Kubernetes system components

- ingress-nginx: Ingress controller

- monitoring: Prometheus, Grafana, Loki

- symbiotic-prod: Application services (production)

- symbiotic-staging: Application services (staging)

Services Deployment:

api-gateway:

replicas: 3

resources:

requests: {cpu: 500m, memory: 512Mi}

limits: {cpu: 1000m, memory: 1Gi}

autoscaling:

minReplicas: 3

maxReplicas: 10

targetCPU: 70%

livenessProbe: /health

readinessProbe: /ready

measurement-service:

replicas: 2

resources:

requests: {cpu: 1000m, memory: 2Gi}

limits: {cpu: 2000m, memory: 4Gi}

autoscaling:

minReplicas: 2

maxReplicas: 5

targetCPU: 75%

policy-service:

replicas: 2

resources:

requests: {cpu: 2000m, memory: 4Gi}

limits: {cpu: 4000m, memory: 8Gi}

autoscaling:

minReplicas: 2

maxReplicas: 5

targetMemory: 80%

nodeAffinity: GPU pool (for local model)

governance-service:

replicas: 2

resources:

requests: {cpu: 500m, memory: 1Gi}

limits: {cpu: 1000m, memory: 2Gi}

frontend (static):

replicas: 3

resources:

requests: {cpu: 100m, memory: 128Mi}

limits: {cpu: 200m, memory: 256Mi}

deployment: nginx serving pre-built React app

timescaledb:

replicas: 1 (MVP), 3 with replication (Production)

resources:

requests: {cpu: 2000m, memory: 8Gi}

limits: {cpu: 4000m, memory: 16Gi}

storage:

class: gp3 (AWS) or pd-ssd (GCP)

size: 500GB (MVP), 2TB+ (Production)

backup:

schedule: Daily at 2 AM UTC

retention: 30 days

destination: S3/GCS

redis:

replicas: 1 (MVP), 3 in cluster mode (Production)

resources:

requests: {cpu: 500m, memory: 2Gi}

limits: {cpu: 1000m, memory: 4Gi}

persistence: RDB snapshots every 5 minutes

Ingress Configuration:

Controller: NGINX Ingress Controller

Rules:

- Host: api.symbiotic.city

Path: /

Service: api-gateway:8000

- Host: app.symbiotic.city

Path: /

Service: frontend:80

- Host: admin.symbiotic.city

Path: /

Service: frontend:80 (different app build)

## TLS:

- Cert-manager (automated Let's Encrypt)

- Wildcard cert: *.symbiotic.city

Annotations:

nginx.ingress.kubernetes.io/rate-limit: "100" (per IP)

nginx.ingress.kubernetes.io/ssl-redirect: "true"

nginx.ingress.kubernetes.io/force-ssl-redirect: "true"

ConfigMaps & Secrets:

ConfigMaps (non-sensitive config):

- app-config: Database URLs, service endpoints

- feature-flags: Enable/disable

Secrets (sensitive data):

- db-credentials: PostgreSQL username/password

- api-keys: External API keys (Claude, Kimi, etc.)

- jwt-secret: Secret for signing JWT tokens

- blockchain-keys: Private keys for smart contract interactions

Secret Management:

- Sealed Secrets OR External Secrets Operator

- Source of truth: HashiCorp Vault

- Automatic rotation (90 days)

- Never commit secrets to Git

Network Policies:

Default: Deny all traffic

Allowed:

- api-gateway → measurement-service, policy-service, governance-service

- measurement-service → timescaledb, redis

- policy-service → timescaledb, redis, external AI APIs

- governance-service → timescaledb, blockchain node

- All services → monitoring (Prometheus)

Egress:

- Allow to external APIs (Anthropic, Moonshot, etc.)

- Allow to blockchain RPC endpoints

- Block everything else by default

Service Mesh (Phase 2):

Technology: Istio OR Linkerd

Benefits:

- mTLS between all services (automatic)

- Observability (request tracing)

- Traffic management (canary deployments)

- Circuit breaking

MVP: Not implemented (keep it simple)

Phase 2: Evaluate if needed

7.3 CI/CD Pipeline

yaml

Version Control:

Platform: GitHub

Repository: symbiotic-codes/iv-technical

Branching Strategy: GitHub Flow

- main: Protected, production-ready

- feature/*: Feature branches

- hotfix/*: Emergency fixes

Branch Protection (main):

- Require PR reviews (2 approvals)

- Require status checks to pass

- No direct commits

- Require linear history

CI Pipeline (GitHub Actions):

.github/workflows/ci.yml:

name: CI Pipeline

on:

push:

branches: [main, develop]

pull_request:

branches: [main]

jobs:

lint:

runs-on: ubuntu-latest

steps:

- Checkout code

- Setup Python 3.11

- Install dependencies

- Run: black --check .

- Run: flake8 .

- Run: mypy .

test:

runs-on: ubuntu-latest

services:

postgres:

image: timescale/timescaledb:latest-pg15

redis:

image: redis:7-alpine

steps:

- Checkout code

- Setup Python 3.11

- Install dependencies

- Run: pytest --cov=. --cov-report=xml

- Upload coverage to Codecov

security-scan:

runs-on: ubuntu-latest

steps:

- Checkout code

- Run: bandit -r . -f json

- Run: safety check

- Run: trivy fs --severity HIGH,CRITICAL .

build-images:

runs-on: ubuntu-latest

if: github.event_name == 'push'

steps:

- Checkout code

- Setup Docker Buildx

- Login to container registry (ECR/GCR)

- Build and push images:

* api-gateway:${{ github.sha }}

* measurement-service:${{ github.sha }}

* policy-service:${{ github.sha }}

* governance-service:${{ github.sha }}

* frontend:${{ github.sha }}

CD Pipeline:

.github/workflows/deploy-staging.yml:

name: Deploy to Staging

on:

push:

branches: [develop]

jobs:

deploy:

runs-on: ubuntu-latest

steps:

- Checkout code

- Setup kubectl

- Configure kubeconfig (staging cluster)

- Update image tags in k8s manifests

- Apply: kubectl apply -f k8s/staging/

- Wait for rollout: kubectl rollout status

- Run smoke tests: curl https://api-staging.symbiotic.city/health

- Notify: Slack webhook

.github/workflows/deploy-production.yml:

name: Deploy to Production

on:

push:

tags: ['v*']  # Triggered by version tags (v1.0.0)

jobs:

deploy:

runs-on: ubuntu-latest

environment: production  # Requires manual approval

steps:

- Checkout code

- Setup kubectl

- Configure kubeconfig (production cluster)

- Blue-Green Deployment:

1. Deploy to "green" environment

2. Run integration tests

3. Switch traffic to "green"

4. Keep "blue" for 1 hour (rollback capability)

5. Decommission "blue"

- Notify: Slack, Email

Deployment Strategies:

Rolling Update (Default):

- Update pods gradually (1-2 at a time)

- Zero downtime

- Automatic rollback on failure

Configuration:

maxSurge: 1         # Max pods above desired count

maxUnavailable: 0   # Always maintain availability

Blue-Green (Production):

- Deploy full new version alongside old

- Switch traffic atomically (via Ingress)

- Easy rollback (switch back)

- Higher resource cost (2x pods temporarily)

Canary (Phase 2):

- Route 5% traffic to new version

- Monitor metrics (error rate, latency)

- Gradually increase: 5% → 25% → 50% → 100%

- Automatic rollback if metrics degrade

Rollback Procedure:

1. Detect issue (monitoring alert OR manual report)

2. Decision: Rollback (if critical) OR fix forward

3. Execute:

- kubectl rollout undo deployment/<name>

- OR switch Ingress to previous version (Blue-Green)

4. Verify: curl health endpoints

5. Investigate root cause

6. Post-mortem within 48 hours

Database Migrations:

Tool: Alembic (Python) OR Flyway

Strategy: Forward-only migrations

Process:

1. Write migration script (up only, no down)

2. Test on staging database

3. Run on production BEFORE deploying app code

4. Ensure backward compatibility (old code works with new schema)

5. Deploy new app code

6. Clean up deprecated columns/tables in next release

Example Migration Workflow:

# Generate migration

alembic revision --autogenerate -m "Add user_preferences table"

# Review generated SQL

# Test on staging

alembic upgrade head  # On staging DB

# Deploy to production

kubectl exec -it postgres-pod -- psql < migration.sql

# Verify

kubectl logs api-gateway-pod | grep "Migration successful"

Monitoring Deployment Health:

Metrics to Watch:

- Error rate (target: <1%)

- Latency (p95 <500ms, p99 <2s)

- CPU/Memory utilization

- Database connection pool

- Blockchain transaction failures

Automated Rollback Triggers:

- Error rate > 5% for 5 minutes

- p99 latency > 5 seconds for 5 minutes

- Pod crash loop (3 restarts in 10 minutes)

Manual Approval Gates:

- Production deployments require TSC approval (2/5)

- Database schema changes require DBA review

- Smart contract deployments require 3/5 multi-sig

7.4 Infrastructure as Code

yaml

Tool: Terraform

Repository Structure:

terraform/

├── modules/

│   ├── kubernetes-cluster/

│   ├── networking/

│   ├── database/

│   ├── monitoring/

│   └── blockchain-node/

├── environments/

```
│   ├── dev/
│   ├── staging/
│   └── production/
└── global/
└── state-backend.tf
```

State Management:

Backend: S3 (AWS) or GCS (Google Cloud)

Locking: DynamoDB (AWS) or GCS with versioning

Encryption: AES-256

Configuration:

```
terraform {
backend "s3" {
bucket = "symbiotic-terraform-state"
key    = "production/terraform.tfstate"
region = "eu-west-1"
encrypt = true
dynamodb_table = "terraform-state-lock"
```

```
}

}
```

Example Module (kubernetes-cluster):

```
# modules/kubernetes-cluster/main.tf

resource "aws_eks_cluster" "main" {

name     = var.cluster_name

role_arn = aws_iam_role.cluster.arn

version  = "1.28"

vpc_config {

subnet_ids              = var.subnet_ids

endpoint_private_access = true

endpoint_public_access  = true

public_access_cidrs     = var.allowed_cidrs

}

encryption_config {

provider {
```

```
key_arn = aws_kms_key.eks.arn

}

resources = ["secrets"]

}

enabled_cluster_log_types = [

"api", "audit", "authenticator"

]

}

resource "aws_eks_node_group" "app" {

cluster_name    = aws_eks_cluster.main.name

node_group_name = "app-nodes"

node_role_arn   = aws_iam_role.node.arn

subnet_ids      = var.private_subnet_ids

scaling_config {

desired_size = var.desired_nodes

max_size     = var.max_nodes

min_size     = var.min_nodes
```

}

instance_types = ["t3.large"]

labels = {

role = "application"

}

}

Deployment Workflow:

1. Make changes to Terraform files

2. Run: terraform plan -out=plan.tfplan

3. Review plan (manual)

4. Apply: terraform apply plan.tfplan

5. Commit .tfstate to remote backend (automatic)

6. Document changes in CHANGELOG.md

Drift Detection:

- Scheduled job (daily): terraform plan

- Alert if drift detected (manual changes outside Terraform)

- Reconcile OR update Terraform to match reality

## VIII. TESTING STRATEGY

8.1 Test Pyramid

yaml

Levels of Testing:

┌─────────────────────────────────────┐

│ Manual Exploratory (5%) │ Pilot city user testing

├─────────────────────────────────────┤

│ E2E / Integration (15%) │ Full user flows

├─────────────────────────────────────┤

│ API / Contract Tests (30%) │ Service boundaries

├─────────────────────────────────────┤

│ Unit Tests (50%) │ Individual functions

└─────────────────────────────────────┘

Unit Tests (50%):

Framework: pytest (Python), Jest (JavaScript)

Coverage Target: >80% for critical modules

Scope:

- Individual functions (pure logic)

- Edge cases and error handling

- Data validation logic

- Calculation algorithms (S(s) components (I,D,E,Em) formulas)

Example:

```
# tests/test_vr_calculator.py

def test_vr_calculation_basic():

data = {

"institutional_transparency": 0.8,

"decision_making_speed": 0.7,

"policy_effectiveness": 0.75,

"citizen_engagement": 0.65

}

result = calculate_vr(data)

assert 0 <= result.value <= 5
```

```
assert result.confidence > 0.5

def test_vr_calculation_missing_data():

data = {"institutional_transparency": 0.8}

result = calculate_vr(data)

assert result.value is None

assert result.confidence < 0.5

def test_vr_calculation_outliers():

data = {

"institutional_transparency": 10.0,  # Invalid

"decision_making_speed": 0.7,

"policy_effectiveness": 0.75,

"citizen_engagement": 0.65

}

with pytest.raises(ValidationError):

calculate_vr(data)
```

Execution:

- Every commit (GitHub Actions)

- Local pre-commit hook

- Coverage report in PR comments

API / Contract Tests (30%):

Framework: pytest + httpx (Python), Supertest (Node.js)

Scope:

- API endpoint responses (schema validation)

- Error handling (4xx, 5xx)

- Authentication and authorization

- Rate limiting

Example:

# tests/test_api_metrics.py

async def test_get_metrics_success(client):

response = await client.get("/v1/metrics/barcelona")

assert response.status_code == 200

data = response.json()

assert "metrics" in data

assert "VR" in data["metrics"]

assert isinstance(data["metrics"]["VR"]["value"], float)

async def test_get_metrics_invalid_city(client):

response = await client.get("/v1/metrics/nonexistent")

assert response.status_code == 404

assert response.json()["type"] == "city_not_found"

async def test_get_metrics_rate_limit(client):

for _ in range(101):  # Exceed limit

await client.get("/v1/metrics/barcelona")

response = await client.get("/v1/metrics/barcelona")

assert response.status_code == 429

Contract Testing (Pact):

- Consumer-driven contracts

- Policy service → AI APIs (Claude, Kimi)

- Frontend → Backend APIs

- Ensures compatibility during independent deployments

Integration Tests (15%):

Framework: pytest + Docker Compose

Scope:

- Multi-service interactions

- Database transactions

- Message queue flows

- Blockchain interactions (testnet)

Example:

```
# tests/integration/test_policy_prediction.py

async def test_policy_prediction_e2e(services):

# Setup: Ensure services running (API Gateway, Policy Service, DB)

# Create policy proposal

response = await services.api_gateway.post(

"/v1/policy/predict",

json={

"policy": {

"title": "Expand bike lanes",
```

```
"description": "Add 50km of protected bike lanes",

"scope": "city",

"horizon": 5

}

},

headers={"Authorization": f"Bearer {test_jwt}"}

)

assert response.status_code == 200

prediction = response.json()

# Verify prediction structure

assert "prediction_id" in prediction

assert "predicted_impact" in prediction

# Verify data persisted to DB

db_record = await services.db.query(

"SELECT * FROM policy_predictions WHERE id = %s",

prediction["prediction_id"]
```

)

assert db_record is not None

# Verify blockchain commitment

assert "blockchain_tx_hash" in prediction

tx = await services.blockchain.get_transaction(

prediction["blockchain_tx_hash"]

)

assert tx.status == "confirmed"

Environment:

- Docker Compose with all services

- Test database (reset between tests)

- Mock external APIs (Claude, Kimi) OR use test API keys

- Polygon Mumbai testnet

E2E Tests (15%):

Framework: Playwright (browser automation)

Scope:

- Critical user journeys (full stack)

- User perspective (black box)

- Cross-browser (Chrome, Firefox, Safari)

Example Flows:

1. User Registration & Onboarding

2. Submit Survey Response

3. View City Dashboard

4. Create Policy Proposal

5. Vote on Governance Proposal

6. Admin Review Queue

Example:

```
# tests/e2e/test_policy_proposal.spec.ts

test('Create and submit policy proposal', async ({ page }) => {

// Login

await page.goto('https://app-staging.symbiotic.city');

await page.click('button:has-text("Connect Wallet")');

// ... MetaMask interaction ...
```

```
// Navigate to proposal creation

await page.click('a:has-text("Governance")');

await page.click('button:has-text("New Proposal")');

// Fill form

await page.fill('input[name="title"]', 'Test Policy');

await page.fill('textarea[name="description"]', 'This is a test policy for E2E
testing...');

await page.selectOption('select[name="scope"]', 'city');

await page.fill('input[name="horizon"]', '5');

// Preview

await page.click('button:has-text("Preview")');

await page.waitForSelector('text=Predicted Impact');

// Verify prediction displayed

const deltaVR = await page.textContent('[data-testid="delta-vr"]');

expect(parseFloat(deltaVR)).toBeGreaterThan(-5);

expect(parseFloat(deltaVR)).toBeLessThan(5);

// Submit

await page.click('button:has-text("Submit Proposal")');
```

```
// Verify success

await page.waitForSelector('text=Proposal submitted successfully');

// Verify appears in proposal list

await page.click('a:has-text("View All Proposals")');

await expect(page.locator('text=Test Policy')).toBeVisible();

});
```

Execution:

- Nightly (full suite on staging)

- Before production deployment (smoke tests)

- On-demand for major releases

Manual Exploratory Testing (5%):

Scope:

- Usability issues

- Edge cases not covered by automated tests

- Visual bugs

- Performance on real devices

Process:

- Test charter (what to explore)

- 30-60 min session

- Document findings in GitHub issues

Focus Areas:

- Mobile app (different devices, OS versions)

- Dashboard visualizations (different data scenarios)

- Accessibility (screen readers, keyboard navigation)

8.2 Test Data Management

yaml

Test Data Strategy:

Synthetic Data:

- Generate realistic test data programmatically

- Faker library (Python) for names, addresses, etc.

- Custom generators for domain-specific data (S(s) components (I,D,E,Em) scores)

Example:

from faker import Faker

```
fake = Faker()

def generate_test_user():

return {

"id": fake.uuid4(),

"email": fake.email(),

"city_id": "barcelona",

"created_at": fake.date_time_this_year()

}
```

Anonymized Production Data:

- Phase 2: Copy production DB to staging

- Anonymize:

* Replace emails with fake@example.com

* Hash user IDs

* Remove sensitive fields

- Preserve statistical properties (for realistic testing)

Test Fixtures:

- Predefined data sets for specific scenarios

- Version controlled (tests/fixtures/)

- JSON or YAML format

Example:

# tests/fixtures/barcelona_baseline.yml

city_id: barcelona

metrics:

## VR:

value: 2.35

confidence: 0.82

breakdown:

institutional_transparency: 0.80

decision_making_speed: 0.70

policy_effectiveness: 0.75

citizen_engagement: 0.65

## EC:

value: 1.78

confidence: 0.75

Data Reset Between Tests:

- Unit tests: No shared state (pure functions)

- Integration tests: Rollback transactions

- E2E tests: Reset database to known state (fixtures)

Implementation:

@pytest.fixture(scope="function")

async def clean_database(db):

# Before test: Load fixtures

await db.load_fixtures("barcelona_baseline.yml")

yield db

# After test: Rollback or truncate

await db.execute("TRUNCATE TABLE metrics_raw CASCADE")

8.3 Performance Testing

yaml

Load Testing:

Tool: Locust (Python) OR k6

Scenarios:

1. Baseline Load:

- 100 concurrent users

- Duration: 10 minutes

- Requests: Mixed (70% reads, 30% writes)

- Target: p95 < 500ms, 0% errors

2. Peak Load:

- 500 concurrent users

- Duration: 5 minutes

- Simulate policy announcement (traffic spike)

- Target: p95 < 1000ms, <1% errors

3. Stress Test:

- Gradually increase users until system breaks

- Identify bottlenecks

- Determine max capacity

4. Endurance Test:

- 50 concurrent users

- Duration: 24 hours

- Check for memory leaks, resource exhaustion

Example Locust Script:

```
from locust import HttpUser, task, between

class SymbioticUser(HttpUser):

wait_time = between(1, 5)

@task(7)  # 70% weight

def get_metrics(self):

self.client.get("/v1/metrics/barcelona")

@task(2)  # 20% weight

def view_proposals(self):

self.client.get("/v1/governance/proposals")

@task(1)  # 10% weight

def predict_policy(self):
```

self.client.post(

"/v1/policy/simulate",

json={"policy": {...}},

headers={"Authorization": f"Bearer {self.token}"}

)

Execution:

- Staging environment (before major releases)

- Automated (nightly) OR on-demand

- Results logged and compared to baseline

Database Performance:

- Query profiling: EXPLAIN ANALYZE on slow queries

- Index optimization: Create indices for frequent queries

- Connection pooling: Tune pool size (default: 20)

- Query timeout: 30 seconds (kill long-running queries)

Monitoring:

- Slow query log (>1 second)

- Connection pool utilization

- Cache hit ratio (target: >90%)

Blockchain Performance:

- Gas optimization: Minimize storage writes

- Batch transactions: Group multiple operations

- Retry logic: Handle network congestion

Metrics:

- Transaction confirmation time (target: <30 seconds)

- Gas cost per transaction (optimize to <$0.10)

- Failed transaction rate (target: <1%)

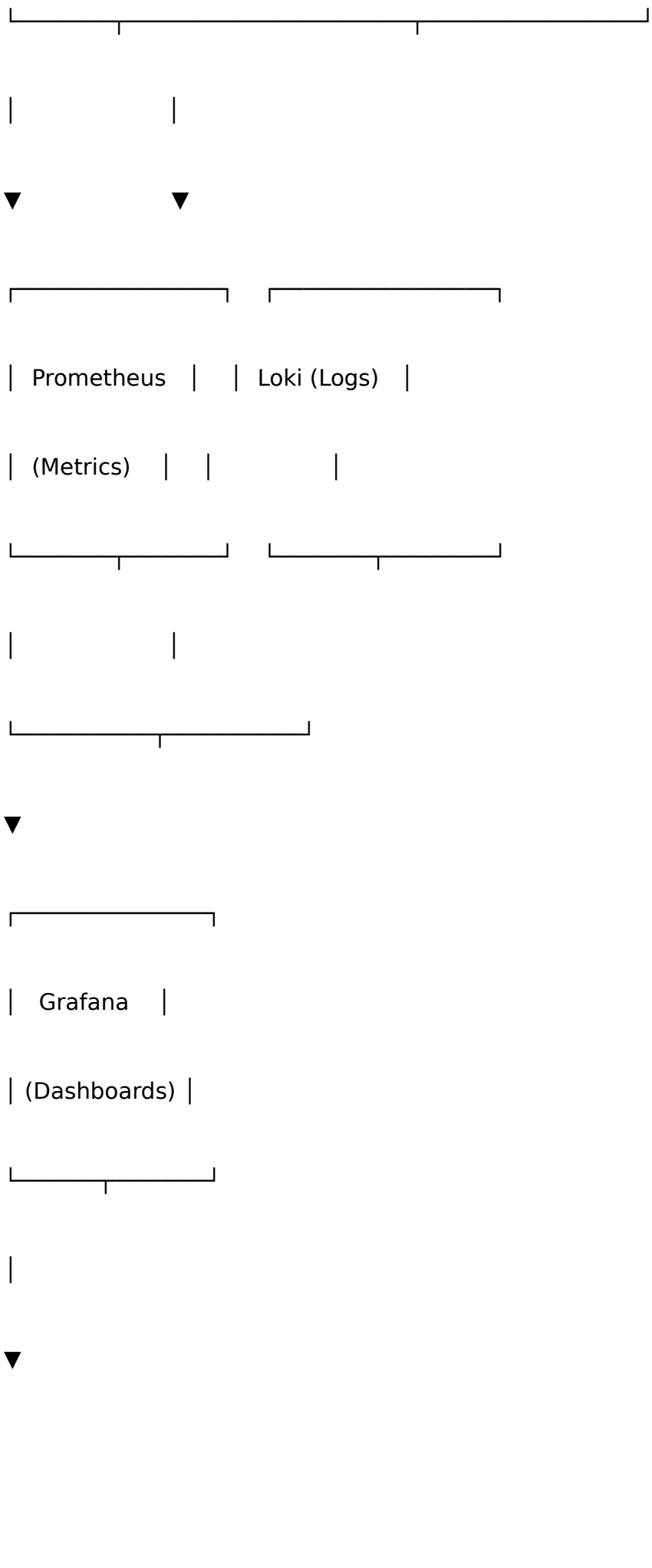
## IX. MONITORING & OPERATIONS

9.1 Observability Stack

yaml

Architecture:

```
┌──────────────────────────────────────────────────┐
│          Application Services              │
│  (API Gateway, Measurement, Policy, etc.)    │
```



┌──────────────────┐

│ Alertmanager │

│ (Alerts) │

└──────────────────┘

Metrics Collection (Prometheus):

Scrape Configuration:

```
global:

scrape_interval: 15s

evaluation_interval: 15s

scrape_configs:

- job_name: 'api-gateway'

kubernetes_sd_configs:

- role: pod

relabel_configs:

- source_labels: [__meta_kubernetes_pod_label_app]

regex: api-gateway

action: keep
```

- job_name: 'measurement-service'

...

- job_name: 'policy-service'

...

- job_name: 'timescaledb'

static_configs:

- targets: ['timescaledb:9187']

- job_name: 'redis'

static_configs:

- targets: ['redis-exporter:9121']

Metrics Exposed by Services:

HTTP Metrics (via middleware):

- http_requests_total (counter)

- http_request_duration_seconds (histogram)

- http_requests_in_progress (gauge)

Business Metrics:

- metrics_calculated_total (counter, labels: city_id, metric_type)

- policy_predictions_total (counter)

- governance_votes_total (counter)

- citizen_opt_in_count (gauge, labels: city_id, opt_in_level)

System Metrics:

- process_cpu_seconds_total

- process_resident_memory_bytes

- go_goroutines (if Go services)

- python_gc_objects_collected_total (if Python)

Custom Metrics:

- vr_score (gauge, labels: city_id)

- ec_score (gauge, labels: city_id)

- bd_score (gauge, labels: city_id)

- policy_ai_latency_seconds (histogram)

- blockchain_tx_confirmation_seconds (histogram)

Retention: 30 days (downsample to long-term storage if needed)

Log Aggregation (Loki):

Log Format: JSON structured logs

Example:

{

"timestamp": "2025-10-21T13:45:23Z",

"level": "INFO",

"service": "api-gateway",

"request_id": "req_a7b3d9f2",

"user_id": "user_uuid",

"method": "POST",

"path": "/v1/policy/predict",

"status": 200,

"duration_ms": 1234,

"message": "Policy prediction completed",

"trace_id": "trace_xyz"  // For distributed tracing

}

Log Levels:

- DEBUG: Verbose (disabled in production)

- INFO: Normal operations

- WARN: Recoverable issues

- ERROR: Errors requiring investigation

- CRITICAL: Service down, immediate action

Retention: 7 days (searchable), 90 days (archived to S3/GCS)

Querying:

- LogQL (Loki's query language)

- Example: {service="policy-service"} |= "error" | json

- Integrated in Grafana dashboards

Distributed Tracing (Optional Phase 2):

Tool: Jaeger OR Tempo

Purpose: Track request across multiple services

Example Flow:

User Request → API Gateway → Policy Service → Claude API

↓

Measurement Service → TimescaleDB

Each hop adds span to trace

Visualize: Gantt chart of request lifecycle

Use Cases:

- Debug slow requests (which service is bottleneck?)

- Identify cascading failures

9.2 Dashboards

yaml

Grafana Dashboards:

1. System Overview (Default Dashboard):

Panels:

- Service Status (UP/DOWN indicators)

- Request Rate (req/s, by service)

- Error Rate (%, by service)

- Latency (p50, p95, p99)

- CPU/Memory Utilization (by pod)

- Database Connections (active/idle)

Refresh: 5 seconds

Audience: Operations team, always visible

2. Business Metrics Dashboard:

Panels:

- Current Metric Scores (VR, EC gauges)

- Metric Trends (time series, last 7/30/90 days)

- Policy Predictions (count, avg confidence)

- Governance Activity (proposals created, votes cast)

- Citizen Engagement (opt-ins, survey completions)

- Biotic Diversity (BD score, alerts)

Refresh: 1 minute

Audience: TSC, city officials, public (filtered)

3. Policy AI Performance Dashboard:

Panels:

- Prediction Latency (histogram)

- Model Agreement (consensus vs disagreement %)

- Confidence Distribution

- Red Flags Triggered (count, by type)

- External API Status (Claude, Kimi uptime)

- Local Model GPU Utilization

Refresh: 30 seconds

Audience: ML engineers, TSC

4. Security Dashboard:

Panels:

- Authentication Failures (rate)

- Rate Limit Hits (by endpoint, IP)

- Suspicious Activity (anomaly detection)

- Certificate Expiry (days remaining)

- Failed Blockchain Transactions

- Data Breach Indicators (based on rules)

Refresh: 10 seconds

Audience: Security team

5. Database Dashboard:

Panels:

- Query Performance (slow queries)

- Table Sizes (disk usage)

- Index Hit Ratio

- Connection Pool Stats

- Replication Lag (if applicable)

- Backup Status (last successful backup)

Refresh: 1 minute

Audience: DBAs, operations

Annotations:

- Deployments (vertical lines marking releases)

- Incidents (highlighted regions)

- Governance votes (markers)

- System changes (parameter updates)

9.3 Alerting

yaml

Alertmanager Configuration:

Alert Rules (Prometheus):

# High-Priority Alerts (P0 - Wake people up)

- alert: ServiceDown

expr: up{job=~"api-gateway|measurement-service|policy-service"} == 0

for: 1m

labels:

severity: critical

annotations:

summary: "Service {{ $labels.job }} is down"

description: "{{ $labels.job }} has been down for 1 minute"

- alert: HighErrorRate

expr: rate(http_requests_total{status=~"5.."}[5m]) / rate(http_requests_total[5m]) > 0.05

for: 5m

labels:

severity: critical

annotations:

```
summary: "High error rate on {{ $labels.service }}"

description: "Error rate is {{ $value | humanizePercentage }}"

- alert: DatabaseDown

expr: up{job="timescaledb"} == 0

for: 1m

labels:

severity: critical

annotations:

summary: "Database is down"

- alert: BioDiversityCritical

expr: bd_score < 0.25

for: 30m

labels:

severity: critical

annotations:

summary: "Biotic Diversity below critical threshold"

description: "BD score is {{ $value }}, threshold is 0.25"
```

# Medium-Priority Alerts (P1 - Business hours response)

- alert: HighLatency

expr: histogram_quantile(0.95, rate(http_request_duration_seconds_bucket[5m])) > 1.0

for: 10m

labels:

severity: warning

annotations:

## X. GOVERNANCE & COMPLIANCE

10.1 Governance Framework

10.1.1 Multi-Stakeholder Structure

Core Principle: Balanced representation across human and non-human interests with explicit power distribution.

Governance Layers:

Layer 1: Human Governance Council

• Composition: 7 voting members:

○ City government representative (1 seat)

○ Civil society organizations (2 seats)

○ Academic/research institutions (1 seat)

○ Private sector (1 seat)

○ Citizens assembly (2 seats - rotating)

• Decision authority:

○ Final veto power on all AI recommendations

○ Budget allocation approval

○ Policy framework definition

○ Emergency override authority

• Meeting frequency: Monthly (minimum), with quarterly strategic reviews

Layer 2: AI Advisory System

• Function: Policy analysis, scenario modeling, trade-off evaluation

• Constraints:

○ Cannot execute decisions autonomously

○ All recommendations require human review

○ Must explain reasoning in human-readable format

○ Subject to algorithmic audits (quarterly)

Layer 3: Biotic Interests Representation

• Mechanism: Formal integration of ecological data into decision-making

• Representation method:

○ Biodiversity metrics treated as "vote" in weighted decisions

○ Decline in BD score triggers mandatory review

○ Ecological thresholds function as "red lines" that cannot be crossed

10.1.2 Decision-Making Protocols

Standard Decisions (routine operations):

• Approval process: Simple majority (4/7 votes)

• Timeline: 5 business days for review

• AI involvement: Recommendation + impact analysis

• Public notice: 48 hours before implementation

Major Decisions (policy changes, budget > $50k, algorithmic changes):

• Approval process: Supermajority (5/7 votes)

• Timeline: 15 business days minimum

• Required steps:

○ Public consultation period (10 days)

○ AI impact assessment

○ Ecological impact review

○ Independent expert review (if requested)

• Public notice: 7 days before implementation

Critical Decisions (fundamental changes, BD score < 0.3, emergency actions):

• Approval process: Unanimous (7/7 votes) OR public referendum

• Timeline: 30 days minimum

• Required steps:

○ Extended public consultation (20 days)

○ Multiple independent reviews

○ Mandatory town hall meetings

○ Full transparency report published

10.1.3 Conflict Resolution Mechanisms

Tier 1: Internal Mediation

• Process: Facilitated discussion within Governance Council

• Timeline: 5 business days

• Success rate target: >80% of conflicts resolved

Tier 2: Expert Panel Review

• Composition: 3 independent experts (ethics, technology, ecology)

• Process: Binding recommendation within 10 days

• Criteria: Alignment with core principles + feasibility

Tier 3: Public Arbitration

• Trigger: Deadlock after Tier 2, or council member request

• Process: Citizens assembly deliberation (50-100 randomly selected residents)

• Timeline: 15-20 days

• Outcome: Binding decision with 60% supermajority

10.2 Ethical Framework

10.2.1 Core Ethical Principles

1. Human Dignity & Autonomy

• Individual rights cannot be overridden by collective optimization

• Citizens retain right to opt-out of AI-driven decisions affecting them personally

• Personal data sovereignty: individuals control their data contributions

2. Transparency & Explainability

• All AI recommendations must include human-readable explanations

• Decision logic must be auditable and reproducible

• "Black box" systems are prohibited in governance applications

3. Ecological Integrity

• Biotic diversity metrics have equal weight to economic metrics

• Long-term ecological health cannot be sacrificed for short-term gains

• Precautionary principle applies to irreversible ecological decisions

4. Equity & Justice

• System benefits must be distributed fairly across socioeconomic groups

• Vulnerable populations receive additional protections

• Historical inequities are acknowledged and addressed in system design

5. Accountability

• Clear responsibility chains for all decisions (human and AI)

• Mandatory incident reporting and analysis

• Public access to governance meeting records and voting histories

10.2.2 Ethical Review Process

Ethics Review Board:

• Composition: 5 members (bioethicist, technologist, legal expert, indigenous representative, youth representative)

• Independence: Members cannot serve on Governance Council simultaneously

• Term: 3 years, staggered to ensure continuity

Review Triggers:

• All new AI algorithms before deployment

• Significant changes to data collection methods

• Decisions with distributional impacts > 10% variance across groups

• Any citizen petition with 100+ signatures

• Annual comprehensive system review

Review Process:

1. Submission: Complete documentation + impact assessment

2. Initial review: 5 business days for completeness check

3. Detailed analysis: 15 days for full ethical evaluation

4. Stakeholder consultation: 10 days for public input

5. Board decision: Approve / Conditional approval / Reject

6. Appeals process: Available within 10 days of decision

10.3 Regulatory Compliance

10.3.1 Data Protection & Privacy

GDPR Compliance (EU):

• Legal basis: Public interest + explicit consent for non-essential data

• Data minimization: Collect only what is necessary for stated purposes

• Right to erasure: Full data deletion within 30 days of request

• Data portability: Export in machine-readable format

• Privacy by design: Technical and organizational measures

• DPO (Data Protection Officer): Required, reports to Governance Council

Anonymization Standards:

• k-anonymity: Minimum k=5 for all aggregated datasets

• Differential privacy: $\varepsilon \leq 1.0$ for statistical queries

• Prohibition on re-identification attempts

• Regular anonymization audits by independent experts

Data Retention Policy:

• Operational data: 2 years unless legally required

• Aggregated analytics: 5 years

• Governance records: 10 years

• Biodiversity data: Indefinite (scientific value)

• Automated deletion schedules with manual review

10.3.2 AI Regulation Compliance

EU AI Act Alignment (High-Risk System Classification):

• Risk assessment: System classified as "high-risk" due to governance application

• Requirements:

○ Risk management system throughout lifecycle

○ Training data governance (quality, bias mitigation)

○ Technical documentation maintained current

○ Automatic logging of operations

○ Human oversight mechanisms

○ Accuracy, robustness, and cybersecurity measures

○ Conformity assessment before deployment

Algorithmic Accountability:

• Model cards: Published for all AI systems (architecture, training data, performance metrics)

• Bias audits: Quarterly assessment across protected characteristics

• Explainability requirements: Local interpretable explanations for individual decisions

• Contestability: Right to challenge AI decisions with human review

10.3.3 Environmental Compliance

Biodiversity Regulations:

• EU Biodiversity Strategy 2030 alignment

• CBD (Convention on Biological Diversity) reporting standards

• Local environmental protection laws compliance

• Required permits for ecological monitoring equipment

Environmental Impact Assessment:

• Initial assessment: Before system deployment

• Monitoring: Continuous ecological impact tracking

• Reporting: Annual environmental report to regulatory authorities

• Mitigation plans: Required for any negative impacts identified

10.4 Risk Management Framework

10.4.1 Risk Categories & Mitigation

Technical Risks:

Risk: AI system failure or erroneous recommendations

• Mitigation: Redundant validation, human-in-the-loop for critical decisions, graceful degradation

• Monitoring: Real-time anomaly detection, weekly model performance reviews

• Contingency: Manual override protocols, fallback to human-only governance

Risk: Data breach or unauthorized access

• Mitigation: Encryption at rest and in transit, zero-trust architecture, regular penetration testing

• Monitoring: 24/7 security operations center, automated threat detection

• Contingency: Incident response plan (<1 hour activation), breach notification (72 hours)

Social Risks:

Risk: Public distrust or rejection of system

• Mitigation: Extensive public education, transparent communication, opt-in participation

• Monitoring: Quarterly public sentiment surveys, continuous stakeholder engagement

• Contingency: Adaptive governance model, ability to scale back or pause deployment

Risk: Exacerbation of existing inequalities

• Mitigation: Equity impact assessments, targeted outreach to vulnerable groups, progressive benefits distribution

• Monitoring: Disaggregated impact analysis by demographic groups

• Contingency: Corrective measures mandate, additional resources for affected communities

Ecological Risks:

Risk: Monitoring infrastructure negatively impacts ecosystems

• Mitigation: Minimal invasive sensors, ecological site assessments, seasonal restrictions

• Monitoring: Independent ecological impact audits (semi-annual)

• Contingency: Equipment relocation or removal protocols

Risk: Biodiversity decline despite system operation

• Mitigation: Conservative BD thresholds, early warning systems, multi-causal analysis

• Monitoring: Real-time BD tracking with weekly trend analysis

• Contingency: Emergency ecology council, mandatory intervention protocols

Governance Risks:

Risk: Capture by special interests or political manipulation

• Mitigation: Multi-stakeholder structure, transparency requirements, conflict of interest policies

• Monitoring: Public audits of decision-making, independent governance assessments

• Contingency: Citizen-initiated governance reviews, recall mechanisms

10.4.2 Incident Management Protocol

Severity Levels:

Severity 1 (Critical): System failure, data breach, safety threat

• Response time: Immediate (< 15 minutes)

• Team: All hands, external experts as needed

• Communication: Public alert within 1 hour, hourly updates

• Post-incident: Full public report within 7 days

Severity 2 (Major): Performance degradation, minor security incident, public complaint

• Response time: < 1 hour

• Team: Technical lead + relevant specialists

• Communication: Stakeholder notification within 4 hours

• Post-incident: Internal review + summary report within 14 days

Severity 3 (Minor): Routine issues, low-impact bugs

• Response time: < 4 hours

• Team: On-call engineer

• Communication: Logged in incident tracking system

• Post-incident: Monthly aggregate review

10.5 Transparency & Public Accountability

10.5.1 Public Reporting Requirements

Quarterly Reports (Public):

• System performance metrics (all four core metrics: VR, EC, VAI, SP)

• Governance Council decisions summary with vote records

• Incident reports (anonymized where appropriate)

• Financial expenditure breakdown

• Biodiversity trends with contextual analysis

• Public engagement summary (consultations, feedback received)

Annual Reports (Comprehensive):

• Full system audit (technical, ethical, environmental)

• Impact assessment across all stakeholder groups

• Lessons learned and adaptive changes implemented

• Strategic roadmap for following year

• Independent evaluator report (external)

• Long-term trend analysis (3-5 year trajectories)

10.5.2 Open Data & API Access

Open Data Portal:

• Aggregated metrics: Available in real-time (anonymized)

• Historical datasets: Full archive with version control

• Formats: CSV, JSON, GeoJSON for spatial data

• License: Creative Commons BY 4.0 (attribution required)

• Update frequency: Daily for operational data, weekly for analytical datasets

Public API:

• Read-only access to non-sensitive aggregated data

• Rate limits: 1000 requests/hour per API key (free tier)

• Authentication: API key required (free registration)

• Documentation: OpenAPI specification, interactive sandbox

• Use cases: Research, third-party visualization, civic tech applications

Restricted Data Access:

• Research applications: Vetted researchers can apply for access to detailed data

• Approval process: Ethics review + data sharing agreement

• Conditions: Secure environment, prohibition on re-identification, publication approval

10.5.3 Public Participation Mechanisms

Citizen Input Channels:

• Online platform: Feedback, questions, and suggestions (response within 5 business days)

• Town halls: Quarterly open meetings (in-person + virtual)

• Focus groups: Targeted engagement with specific communities

• Surveys: Annual comprehensive assessment of public perception

• Citizen petitions: 100 signatures trigger formal review

Participatory Budgeting:

• Allocation: 10% of annual budget ($45k in Year 1) decided by public vote

• Process: Proposal submission → feasibility review → public vote

• Eligibility: All city residents, proposals align with system values

• Timeline: Annual cycle (Q4 proposals, Q1 voting, Q2-Q4 implementation)

Citizens Assembly:

• Selection: 50 residents via civic lottery (stratified random sampling)

• Rotation: New assembly every 6 months (2 seats on Governance Council)

• Training: 2-day orientation on system, ethics, governance

• Compensation: Stipend for time commitment (~10 hours/month)

• Authority: Advisory role + 2 voting seats on Council

10.6 Audit & Evaluation Framework

10.6.1 Internal Audits

Technical Audits:

• Frequency: Quarterly

• Scope: Code review, algorithm performance, data quality, security posture

• Auditor: Internal technical team + rotating external expert

• Output: Technical audit report with recommendations

Financial Audits:

• Frequency: Annual

• Scope: Budget compliance, expenditure verification, value-for-money assessment

• Auditor: Independent certified public accountant

• Output: Financial audit report (publicly available)

Operational Audits:

• Frequency: Semi-annual

• Scope: Process compliance, SLA adherence, incident management effectiveness

• Auditor: Operations manager + governance representative

• Output: Operational improvement plan

10.6.2 External Evaluations

Independent Impact Evaluation:

• Frequency: Annual

• Evaluator: Third-party research organization (selected via RFP)

• Methodology: Mixed methods (quantitative metrics + qualitative assessment)

• Focus areas:

○ Goal achievement (metric targets vs. actuals)

○ Social impact (equity, inclusion, public trust)

○ Ecological outcomes (BD trends, causal attribution)

○ Governance effectiveness

○ Unintended consequences

• Output: Public evaluation report with actionable recommendations

Peer Review:

• Frequency: Biennial

• Reviewers: International experts in AI governance, urban systems, ecology

• Process: Site visit (3-5 days), stakeholder interviews, data review

• Output: Peer review report with comparative analysis to similar initiatives

Algorithmic Audits:

• Frequency: Quarterly

• Auditor: AI ethics specialist with technical expertise

• Focus: Bias detection, fairness metrics, explainability assessment

• Methodology: Adversarial testing, counterfactual analysis, subgroup performance

• Output: Algorithmic audit report with mitigation strategies

## XI. IMPLEMENTATION ROADMAP

11.1 Roadmap Overview

Duration: 12 months (December 2025 - November 2026)

Phases: 4 quarters with defined milestones and deliverables

Approach: Agile methodology with 2-week sprints, monthly releases, quarterly strategic reviews

Success Criteria: Functional MVP deployed in pilot city with measurable impact on all 4 core metrics

11.2 Implementation Phases

11.2.1 Q1 (Dec 2025 - Feb 2026): Foundation Phase

Objectives:

• Establish governance structures

• Build core technical infrastructure

• Begin stakeholder engagement

• Deploy initial sensing layer

Key Activities:

Month 1 (December 2025):

• Governance Council formation (member selection + onboarding)

• Ethics Review Board establishment

• Core team recruitment (technical architect, data scientists)

• Infrastructure setup: Cloud accounts (AWS/GCP), CI/CD pipelines, dev environments

• Stakeholder mapping and initial outreach

Month 2 (January 2026):

• Database architecture implementation (TimescaleDB + PostGIS setup)

• API framework development (FastAPI skeleton + authentication)

• Biodiversity sensor procurement and site assessment

• Public kickoff event (200+ attendees target)

• First Governance Council meeting

Month 3 (February 2026):

• Layer 1 (Sensing) deployment: 10 biodiversity monitoring stations operational

• Data ingestion pipelines (IoT → TimescaleDB)

• Basic visualization dashboard (Grafana)

• Initial data collection begins

• Citizens Assembly selection (first cohort)

Q1 Deliverables:

✓ Governance structures operational

✓ Core infrastructure deployed

✓ 10 biodiversity sensors active

✓ Data flowing into central system

✓ Public engagement initiated

Q1 Budget: $125,000 (28% of total)

11.2.2 Q2 (Mar - May 2026): Core Development Phase

Objectives:

• Build analytics and AI layers

• Develop policy recommendation engine (prototype)

• Expand sensor network

• Launch public data portal

Month 4 (March 2026):

• Layer 2 (Analytics) development: BD calculation engine, trend analysis

• Historical data integration (existing city datasets)

• API v1 endpoints (read-only public access)

• First quarterly public report published

Month 5 (April 2026):

• Layer 3 (Policy AI) prototype: Scenario modeling, trade-off analysis

• Explainability framework implementation (SHAP, LIME)

• Sensor expansion: 20 additional stations (total: 30)

• Public data portal launch (open data + API documentation)

Month 6 (May 2026):

• Policy AI testing: 5 test scenarios evaluated by Governance Council

• Human oversight interface (dashboard for Council)

• First algorithmic audit (independent)

• Mid-project evaluation (internal)

Q2 Deliverables:

✓ Analytics engine operational

✓ Policy AI prototype functional

✓ 30 sensors deployed

✓ Public API and data portal live

✓ First policy recommendations generated

Q2 Budget: $135,000 (30% of total)

11.2.3 Q3 (Jun - Aug 2026): Integration & Testing Phase

Objectives:

• Complete full-stack integration

• Deploy governance layer

• Conduct comprehensive testing

• Implement temporal weighting (future bias)

Month 7 (June 2026):

• Layer 4 (Governance) implementation: DAO smart contracts (basic), voting mechanisms

• Integration testing: All layers communicating correctly

• Security audit (penetration testing)

• Performance optimization (latency < 200ms target)

Month 8 (July 2026):

• Layer 5 (Regeneration) development: Temporal weight calculator, future scenarios

• User acceptance testing (UAT): Governance Council + Citizens Assembly

• Stress testing (simulated load)

• Documentation completion (technical + user guides)

Month 9 (August 2026):

• Bug fixing and refinement

• Training programs: Staff, Council members, public workshops

• Pre-launch public campaign (awareness + education)

• Disaster recovery and backup testing

Q3 Deliverables:

✓ All 5 layers integrated and tested

✓ Governance mechanisms operational

✓ Security and performance validated

✓ Team and public trained

✓ System ready for production launch

Q3 Budget: $110,000 (24% of total)

11.2.4 Q4 (Sep - Nov 2026): Launch & Optimization Phase

Objectives:

• Official public launch

• Monitor real-world performance

• Gather feedback and iterate

• Measure initial impact on core metrics

Month 10 (September 2026):

• Public launch event (MVP goes live)

• First production decision cycle (Council uses AI recommendations)

• 24/7 monitoring and support begins

• Media engagement and press coverage

Month 11 (October 2026):

• Performance monitoring: Real-time metrics tracking, user behavior analysis

• Iterative improvements: Bug fixes, UX enhancements based on feedback

• First impact assessment: Changes in BD score, public engagement levels

• Academic partnership: Begin collaboration with research institutions

Month 12 (November 2026):

• First annual evaluation (external evaluator)

• Comprehensive annual report (published)

• Lessons learned workshop (team + stakeholders)

• Phase 2 planning: Roadmap for next 12-24 months

Q4 Deliverables:

✓ MVP successfully launched and operational

✓ Governance decisions informed by AI recommendations

✓ Measurable impact on at least 2/4 core metrics

✓ Public trust and engagement maintained/growing

✓ Phase 2 roadmap defined

Q4 Budget: $80,000 (18% of total)

11.3 Critical Path & Dependencies

Critical Path Items:

1. Governance Council formation → ALL subsequent decisions depend on this

2. Database architecture → Must be complete before sensor deployment

3. Data ingestion pipelines → Required before analytics layer

4. Analytics engine → Required before Policy AI

5. Policy AI prototype → Required for governance layer testing

External Dependencies:

• City government approvals (site access for sensors, data sharing agreements)

• Environmental permits (for sensor deployment in protected areas)

• Hardware procurement lead times (sensors, servers - assume 4-6 weeks)

• Third-party services (cloud providers, external auditors)

• Academic partnerships (for evaluation and validation)

Risk Mitigation for Critical Path:

• Buffer time: 2-week contingency in each phase

• Parallel workflows: Non-dependent tasks proceed simultaneously

• Early procurement: Hardware orders placed in Month 1

• Stakeholder alignment: Weekly check-ins with key dependencies

11.4 Success Metrics by Phase

Q1 Success Criteria:

• Governance Council: 7 members confirmed, first 3 meetings held

• Infrastructure: 95%+ uptime for core systems

• Sensors: 10 stations operational, data quality >90%

• Engagement: 200+ attendees at kickoff, 50+ active participants

Q2 Success Criteria:

• Analytics: BD score calculation accurate within ±5%

• Policy AI: 5 test scenarios evaluated, Council satisfaction >70%

• API: 100+ registered users, 1000+ daily requests

• Sensors: 30 stations, <5% downtime

Q3 Success Criteria:

• Integration: All layers communicating, <1% error rate

• Testing: 95%+ test coverage, zero critical bugs

• Security: Pass external audit, zero vulnerabilities (critical/high)

• Training: 50+ people trained, >80% competency scores

Q4 Success Criteria:

• Launch: System live, 99%+ uptime

• Decisions: 5+ Council decisions informed by AI

• Impact: Measurable improvement in 2/4 core metrics

• Trust: Public approval >60% (survey)

11.5 Resource Allocation

11.5.1 Budget Breakdown

Total Budget: $450,000

Personnel (60% - $270,000):

• Technical Architect: $80,000 (1 FTE, 12 months)

• Data Scientists: $100,000 (2 FTE, 12 months)

• Full-Stack Developers: $60,000 (2 FTE, 12 months)

• Community Engagement Specialist: $30,000 (1 FTE, 12 months)

Infrastructure (20% - $90,000):

• Cloud services (AWS/GCP): $45,000

• Biodiversity sensors (30 units @ $1000): $30,000

• IoT connectivity (cellular/LoRa): $10,000

• Security tools and services: $5,000

External Services (12% - $54,000):

• Independent evaluator: $20,000

• Security audits: $15,000

• Algorithmic audits: $10,000

• Legal/compliance consulting: $9,000

Community Engagement (5% - $22,500):

• Events (kickoff, town halls, workshops): $12,000

• Materials (printed, digital): $5,000

• Citizens Assembly stipends: $5,500

Contingency (3% - $13,500):

• Unforeseen expenses, cost overruns

11.5.2 Team Structure & Roles

Core Technical Team (6 FTE):

Technical Architect (1 FTE):

• System design and architectural decisions

• Technical leadership and code reviews

• Integration oversight

Data Scientists (2 FTE):

• Biodiversity analytics and modeling

• Policy AI development

• Algorithm optimization and validation

Full-Stack Developers (2 FTE):

• API development (backend)

• Dashboard and visualization (frontend)

• Database management

Community Engagement Specialist (1 FTE):

• Stakeholder communication

• Public events organization

• Feedback collection and analysis

Pilot City Team (3 part-time):

• City liaison (0.5 FTE): Coordination with municipal government

• Ecology expert (0.5 FTE): Biodiversity monitoring guidance

• Operations coordinator (0.5 FTE): Day-to-day logistics

11.6 Risk Register

High-Priority Risks:

Risk 1: Delays in governance structure formation

• Impact: Cascading delays across entire project

• Probability: Medium (30%)

• Mitigation: Early outreach, incentives for participation, backup candidates

• Contingency: Interim advisory board if full Council not ready

Risk 2: Technical complexity exceeds estimates

• Impact: Budget overrun, timeline extension

• Probability: Medium (25%)

• Mitigation: Phased development, agile methodology, regular reviews

• Contingency: Scope reduction (defer some features to Phase 2)

Risk 3: Public resistance or skepticism

• Impact: Low participation, political pressure to cancel

• Probability: Low (15%)

• Mitigation: Transparent communication, inclusive design, education campaigns

• Contingency: Adaptive engagement strategy, pilot-within-pilot (smaller scale)

Risk 4: Data quality issues from sensors

• Impact: Inaccurate BD scores, loss of credibility

• Probability: Medium (20%)

• Mitigation: Sensor redundancy, calibration protocols, data validation pipelines

• Contingency: Manual data collection backup, partnership with existing monitoring programs

11.7 Phase 2 Preview (Post-MVP)

Timeline: 12-24 months after MVP launch (Dec 2026 - Nov 2028)

Objectives:

• Scale to additional cities (2-3 new pilots)

• Advanced AI capabilities (predictive modeling, optimization)

• Enhanced governance mechanisms (fully operational DAO)

• Integration with existing urban systems

• International knowledge sharing network

Key Enhancements:

• Predictive analytics: 6-12 month forecasting for BD trends

• Multi-objective optimization: Balance competing priorities algorithmically

• Citizen science integration: Crowdsourced data collection

• Cross-city benchmarking: Comparative performance analysis

• Mobile applications: Broader public participation

Success Conditions for Phase 2:

• MVP operational for ≥6 months with 99%+ uptime

• Demonstrable impact on all 4 core metrics

• Public trust ≥70% (sustained)

• Additional funding secured ($800k-$1.2M estimated)

• Governance Council endorsement for expansion

## XII. APPENDICES

12.1 Glossary of Terms

API (Application Programming Interface): A set of protocols and tools that allows different software applications to communicate with each other.

Biotic Diversity (BD): A composite metric measuring the variety and abundance of living organisms in an ecosystem, including species richness, genetic diversity, and ecosystem complexity.

DAO (Decentralized Autonomous Organization): A blockchain-based organization governed by smart contracts and collective decision-making rather than centralized authority.

Differential Privacy: A mathematical framework that adds controlled noise to data to protect individual privacy while enabling statistical analysis.

DXA (Drawing Units): Measurement unit in Word documents; 1440 DXA = 1 inch.

Evolutionary Coherence (EC): The degree of alignment between genetic, cultural, and technological codes in a civilization's development.

GDPR: General Data Protection Regulation - EU law governing data protection and privacy.

k-anonymity: A privacy technique ensuring each individual is indistinguishable from at least k-1 others in a dataset.

LIME (Local Interpretable Model-agnostic Explanations): A technique for explaining individual predictions of any machine learning model.

MVP (Minimum Viable Product): The simplest version of a system with core features sufficient to test and validate the concept.

PostGIS: A spatial database extension for PostgreSQL that enables geographic data storage and queries.

SHAP (SHapley Additive exPlanations): A method for explaining machine learning model predictions based on game theory.

Symbiotic AI (SAI): Artificial intelligence designed to work in mutually beneficial partnership with humans and natural systems.

Temporal Weight: A multiplier applied to decisions based on their long-term consequences, giving greater weight to impacts on future generations.

TimescaleDB: A time-series database built on PostgreSQL, optimized for handling time-stamped data.

Vector Resilience (VR): A civilization's ability to maintain its developmental trajectory despite external disruptions and internal challenges.

## 12.2 References & Further Reading

Technical Standards:

- ISO/IEC 27001: Information Security Management
- ISO 14001: Environmental Management Systems
- IEEE 7010: Well-being Metrics for Ethical AI
- W3C Web Content Accessibility Guidelines (WCAG) 2.1

Regulatory Frameworks:

- EU AI Act (2024) - https://artificialintelligenceact.eu/
- GDPR Official Text - https://gdpr.eu/

• EU Biodiversity Strategy 2030 - https://environment.ec.europa.eu/strategy/biodiversity-strategy-2030_en

• Convention on Biological Diversity - https://www.cbd.int/

Academic Literature:

• Symbiotic Codes: Vector of Creation (Conceptual Framework) - Original work

• Ostrom, E. (1990). Governing the Commons - Commons governance principles

• Meadows, D. (2008). Thinking in Systems - Systems thinking fundamentals

• Russell, S. (2019). Human Compatible AI - AI alignment research

• Wilson, E.O. (2016). Half-Earth - Biodiversity conservation strategy

Technical Documentation:

• FastAPI Documentation - https://fastapi.tiangolo.com/

• TimescaleDB Documentation - https://docs.timescale.com/

• React Documentation - https://react.dev/

• Prometheus Monitoring - https://prometheus.io/docs/

12.3 Acronyms & Abbreviations

AI: Artificial Intelligence

API: Application Programming Interface

BD: Biotic Diversity

CBD: Convention on Biological Diversity

CI/CD: Continuous Integration / Continuous Deployment

DAO: Decentralized Autonomous Organization

DPO: Data Protection Officer

EC: Evolutionary Coherence

FTE: Full-Time Equivalent

GDPR: General Data Protection Regulation

IoT: Internet of Things

JSON: JavaScript Object Notation

LIME: Local Interpretable Model-agnostic Explanations

MVP: Minimum Viable Product

REST: Representational State Transfer

SAI: Symbiotic AI

SHAP: SHapley Additive exPlanations

TDD: Technical Design Document

UAT: User Acceptance Testing

VR: Vector Resilience

12.4 Sample Data Schemas

Biodiversity Observation Schema (JSON):


```
{
  "observation_id": "bd_obs_001",
  "timestamp": "2026-03-15T14:30:00Z",
  "location": {
    "latitude": 41.3851,
    "longitude": 2.1734,
    "site_id": "BCN_PARK_001"
  },
  "sensor_data": {
    "temperature_c": 18.5,
    "humidity_pct": 65,
    "sound_level_db": 42
  },
  "species_detected": [
    {"species": "Parus major", "confidence": 0.92}
  ],
  "calculated_metrics": {
    "local_bd_score": 0.45
  }
}
```


12.5 Contact Information & Support

Project Website: https://symbiotic-codes.org

Documentation: https://docs.symbiotic-codes.org

Contact Channels:

• General: info@symbiotic-codes.org

• Technical: support@symbiotic-codes.org

• Security: security@symbiotic-codes.org

12.6 Document Version History

Version 1.0 (October 22, 2025):

• Initial complete draft - all 12 sections

• Based on IV.2 Unified Technical Design

• Status: DRAFT - Pending stakeholder review

12.7 Acknowledgments

This Technical Design Document represents collaborative work across multiple domains:

Conceptual Foundation:

• Original Symbiotic Codes framework

• Multi-AI validation and refinement

• Civilizational maturity metrics integration

Special Recognition:

This project represents a unique experiment in human-AI collaborative design for civilizational-scale challenges.

## ---  END OF DOCUMENT  ---

## SYMBIOTIC CODES IV.3

Technical Design Document v1.0

October 22, 2025